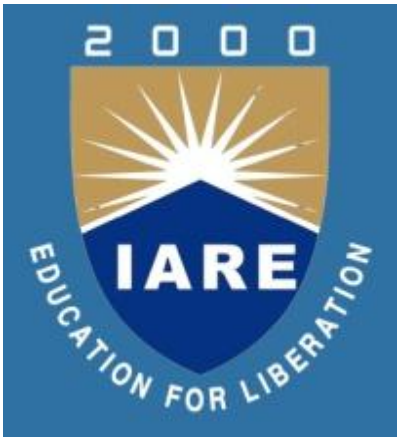
LECTURE NOTES ON

ADVANCED COMPUTER AIDED DESIGN

M.Tech: CAD / CAM - I sem

by
Dr. K RAGHU RAM MOHAN REDDY
Professor



Department of Mechanical Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal – 500 043, Hyderabad

UNIT- I

What is computer Graphics?

Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of programming. Computer graphics image is made up of number of pixels. **Pixel** is the smallest addressable graphical unit represented on the computer screen.

Introduction

Computer is information processing machine. User needs to communicate with computer and the computer graphics is one of the most effective and commonly used ways of communication with the user.

- It displays the information in the form of graphical objects such as pictures, charts, diagram and graphs.
- Graphical objects convey more information in less time and easily understandable formats for example statically graph shown in stock exchange.
- In computer graphics picture or graphics objects are presented as a collection of discrete pixels.
- We can control intensity and color of pixel which decide how picture look like.
- The special procedure determines which pixel will provide the best approximation to the desired picture or graphics object this process is known as **Rasterization.**
- The process of representing continuous picture or graphics object as a collection of discrete pixels is called **Scan Conversion.**

Advantages of computer graphics

- Computer graphics is one of the most effective and commonly used ways of communication with computer.
- It provides tools for producing picture of "real-world" as well as synthetic objects such as mathematical
  surfaces in 4D and of data that have no inherent geometry such as survey result.
- It has ability to show moving pictures thus possible to produce animations with computer graphics.
- With the use of computer graphics we can control the animation by adjusting the speed, portion of picture in view the amount of detail shown and so on.
- It provides tools called motion dynamics. In which user can move objects as well as observes as per requirement for example walk throw made by builder to show flat interior and surrounding.
- It provides facility called update dynamics. With this we can change the shape color and

other properties of object.

- Now in recent development of digital signal processing and audio synthesis chip the interactive graphics can now provide audio feedback along with the graphical feed backs.

Application of computer graphics

- User interface: - Visual object which we observe on screen which communicates with user is one of the most useful applications of the computer graphics.
- Plotting of graphics and chart in industry, business, government and educational organizations drawing like bars, pie-charts, histogram's are very useful for quick and good decision making.
- Office automation and desktop publishing: - It is used for creation and dissemination of information. It is used in in-house creation and printing of documents which contains text, tables, graphs and other forms of drawn or scanned images or picture.
- Computer aided drafting and design: - It uses graphics to design components and system such as automobile bodies structures of building etc.
- Simulation and animation: - Use of graphics in simulation makes mathematic models and mechanical systems more realistic and easy to study.
- Art and commerce: - There are many tools provided by graphics which allows used to make their picture animated and attracted which are used in advertising.
- Process control: - Now a day's automation is used which is graphically displayed on the screen.
- Cartography: - Computer graphics is also used to represent geographic maps, weather maps, oceanographic charts etc.
- Education and training: - Computer graphics can be used to generate models of physical, financial and economic systems. These models can be used as educational aids.
- Image processing: - It is used to process image by changing property of the image.
-

Difference between random scan and raster scan

Base of Difference	**Raster Scan System**	**Random Scan System**
Electron Beam	The electron beam is swept across the screen, one row at a time, from top to bottom.	The electron beam is directed only to the parts of screen where a picture is to be drawn.

Resolution	Its resolution is poor because raster system in contrast produces zigzag lines that are plotted as discrete point sets.	Its resolution is good because this system produces smooth lines drawings because CRT beam directly follows the line path.
Picture Definition	Picture definition is stored as a set of intensity values for all screen points, called pixels in a refresh buffer area.	Picture definition is stored as a set of line drawing instructions in a display file.
Realistic Display	The capability of this system to store intensity values for pixel makes it well suited for the realistic display of scenes contain shadow and color pattern.	These systems are designed for line-drawing and can't display realistic shaded scenes.
Draw an Image	Screen points/pixels are used to draw an image.	Mathematical functions are used to draw an image.

Graphics software and standard

- There are mainly two types of graphics software:
  1. General programming package
  2. Special-purpose application package

General programming package

- A general programming package provides an extensive set of graphics function that can be used in high level programming language such as C or FORTRAN.
- It includes basic drawing element shape like line, curves, polygon, color of element transformation etc.
- Example: - GL (Graphics Library).

Special-purpose application package

- Special-purpose application package are customize for particular application which implement required facility and provides interface so that user need not to vory about how it will work (programming). User can simply use it by interfacing with application.
- Example: - CAD, medical and business systems.

Coordinate representations

- Except few all other general packages are designed to be used with Cartesian coordinate

specifications.

- If coordinate values for a picture are specified is some other reference frame they must be converted to Cartesian coordinate before giving input to graphics package.
- Special-purpose package may allow use of other coordinates which suits application.
- In general several different Cartesian reference frames are used to construct and display scene.
- We can construct shape of object with separate coordinate system called modeling coordinates or sometimes local coordinates or master coordinates.
- Once individual object shapes have been specified we can place the objects into appropriate positions called world coordinates.
- Finally the World-coordinates description of the scene is transferred to one or more output device reference frame for display. These display coordinates system are referred to as "**Device Coordinates"** or "**Screen Coordinates".**
- Generally a graphic system first converts the world-coordinates position to normalized device coordinates. In the range from 0 to 1 before final conversion to specific device coordinates.
- An initial modeling coordinates position (Xmc,Ymc) in this illustration is transferred to a device coordinates position(Xdc,Ydc) with the sequence (Xmc,Ymc) (Xwc,Ywc) (Xnc,Ync) (Xdc,Ydc).

Graphic Function

- A general purpose graphics package provides user with Varity of function for creating and manipulating pictures.
- The basic building blocks for pictures are referred to as output primitives. They includes character, string, and geometry entities such as point, straight lines, curved lines, filled areas and shapes defined with arrays of color points.
- Input functions are used for control & process the various input device such as mouse, tablet, etc.
- Control operations are used to controlling and housekeeping tasks such as clearing display screen etc.
- All such inbuilt function which we can use for our purpose are known as graphics function

Points and Lines

- Point plotting is done by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use.
- Line drawing is done by calculating intermediate positions along the line path between two specified endpoint positions.
- The output device is then directed to fill in those positions between the end points with some color.
- For some device such as a pen plotter or random scan display, a straight line can be drawn smoothly from one end point to other.

- Digital devices display a straight line segment by plotting discrete points between the two endpoints.
- Discrete coordinate positions along the line path are calculated from the equation of the line.
- For a raster video display, the line intensity is loaded in frame buffer at the corresponding pixel positions.
- Reading from the frame buffer, the video controller then plots the screen pixels.
- Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints.
- For example line position of (12.36, 23.87) would be converted to pixel position (12, 24).

Bresenham's Line Algorithm

- An accurate and efficient raster line-generating algorithm, developed by Bresenham which scan converts line using only incremental integer calculations that can be modified to display circles and other curves.
- Figure shows section of display screen where straight line segments are to be drawn.

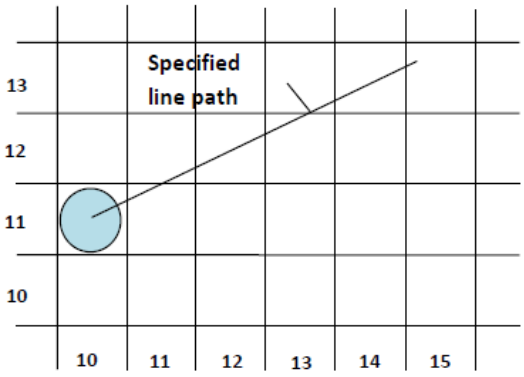


Fig. 2.4: - Section of a display screen where a straight line segment is to be plotted, starting from the pixel at column 10 on scan line 11.

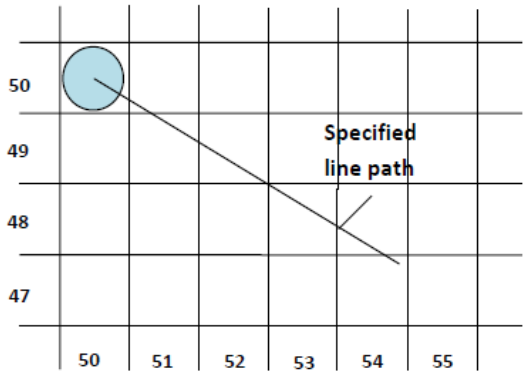


Fig. 2.5: - Section of a display screen where a negative slope line segment is to be plotted, starting from the pixel at column 50 on scan line 50.

- The vertical axes show scan-line positions and the horizontal axes identify pixel column.
- Sampling at unit x intervals in these examples, we need to decide which of two possible pixel position is closer to the line path at each sample step.
- To illustrate bresenham's approach, we first consider the scan-conversion process for lines with positive slope less than 1.
- Pixel positions along a line path are then determined by sampling at unit x intervals.
- Starting from left endpoint (x_0, y_0) of a given line, we step to each successive column and plot the pixel whose scan-line y values is closest to the line path.
- Assuming we have determined that the pixel at (x_k, y_k) is to be displayed, we next need to decide which pixel to plot in column $x_k + 1$.
- Our choices are the pixels at positions $(x_k + 1, y_k)$ and $(x_k + 1, y_k + 1)$.
- Let's see mathematical calculation used to decide which pixel position is light up.
- We know that equation of line is:

  $y = mx + b$

  Now for position $x_k + 1$.

  $y = m(x_k + 1) + b$
- Now calculate distance bet actual line's y value and lower pixel as d_1 and distance bet actual line's y value and upper pixel as d_2.

  $d_1 = y - y_k$

  $\mathrm{d_1 = m(x_k + 1) + b - y_k}$ .....(1)

  $d_2 = (y_k + 1) - y$

  $d_2 = (y_k + 1) - m(x_k + 1) - b$ .....(2)
- Now calculate $d_1 - d_2$ from equation (1) and (2).

  $d_1 - d_2 = (y - y_k) - ((y_k + 1) - y)$

  $d_1 - d_2 = \{m(x_k + 1) + b - y_k\} - \{(y_k + 1) - m(x_k + 1) - b\}$

  $d_1 - d_2 = \{mx_k + m + b - y_k\} - \{y_k + 1 - mx_k - m - b\}$

  $d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1$ .....(3)
- Now substitute $m = \Delta y/\Delta x$ in equation (3)

  $d_1 - d_2 = 2\left(\frac{\Delta y}{\Delta x}\right)(x_k + 1) - 2y_k + 2b - 1$ .....(4)
- Now we have decision parameter p_k for k^{th} step in the line algorithm is given by:

  $p_k = \Delta x(d_1 - d_2)$

  $p_k = \Delta x(2\Delta y/\Delta x(x_k + 1) - 2y_k + 2b - 1)$

  $p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x b - \Delta x$

  $p_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + 2\Delta x b - \Delta x$ .....(5)

  $p_k = 2\Delta y x_k - 2\Delta x y_k + C\ (Where\ Constant\ C = 2\Delta y + 2\Delta x b - \Delta x)$.....(6)

- The sign of p_k is the same as the sign of $d_1 - d_2$, since $\Delta x > 0$ for our example.
- Parameter c is constant which is independent of pixel position and will eliminate in the recursive calculation for p_k.
- Now if p_k is negative then we plot the lower pixel otherwise we plot the upper pixel.
- So successive decision parameters using incremental integer calculation as:
  $p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + \mathrm{C}$
- Now Subtract pk from p_{k+1}
  $p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k)$ -$2\Delta x(y_{k+1} - y_k)$
  $p_{k+1} - p_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + \mathrm{C} - 2\Delta y x_k + 2\Delta x y_k - \mathrm{C}$
  But $x_{k+1} = x_k + 1$, so that $(x_{k+1} - x_k) = 1$
  $p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$
- Where the terms $y_{k+1} - y_k$ is either 0 or 1, depends on the sign of parameter p_k.
- This recursive calculation of decision parameters is performed at each integer x position starting at the left coordinate endpoint of the line.
- The first decision parameter p_0 is calculated using equation (5) as first time we need to take constant part into account so:
  $p_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + 2\Delta x b - \Delta x$
  $p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x b - \Delta x$

  $Now\ Substitute\ m = \Delta y/\Delta x$
  $p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x(y_0 - (\Delta y/\Delta x)x_0) - \Delta \mathrm{x}$
  $p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta \mathrm{x}$
  $p_0 = 2\Delta y - \Delta \mathrm{x}$
- Let's see Bresenham's line drawing algorithm for $|m| < 1$
  1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
  2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
  3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

4. At each x_k along the line, starting at $k = 0$, perform the following test:

   If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

   $$p_{k+1} = p_k + 2\Delta y$$

   Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

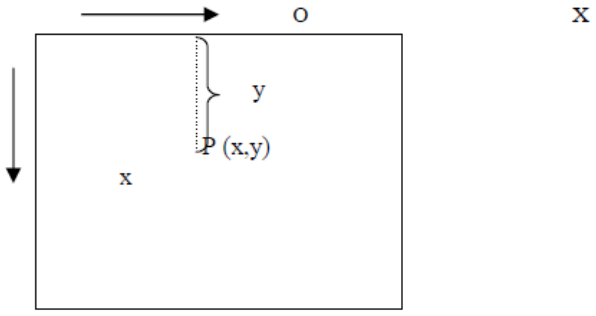
   $$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step-4 Δx times.

- Bresenham's algorithm is generalized to lines with arbitrary slope by considering symmetry between the various octants and quadrants of the xy plane.
- For lines with positive slope greater than 1 we interchange the roles of the x and y directions.
- Also we can revise algorithm to draw line from right endpoint to left endpoint, both x and y decrease as we step from right to left.
- When $d_1 - d_2 = 0$ we choose either lower or upper pixel but once we choose lower than for all such case for that line choose lower and if we choose upper the for all such case choose upper.
- For the negative slope the procedure are similar except that now one coordinate decreases as the other increases.
- The special case handle separately. Horizontal line ($\Delta y = 0$), vertical line ($\Delta x = 0$) and diagonal line with $|\Delta x| = |\Delta y|$ each can be loaded directly into the frame buffer without processing them through the line plotting algorithm.

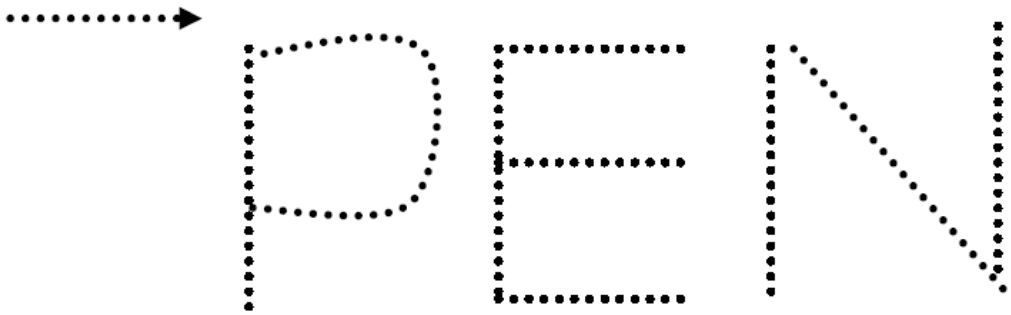
Point Plotting Techniques

The coordinate systems of the monitor: Point plotting techniques are based on the use of Cartesian coordinate system. Each point is addressed by two points (x,y) which indicate the distance of the point with respect to the origin. p (x,y) is pixel at a horizontal distance x and vertical distance y from the origin



Now any picture to be displayed is to be represented as a combination of points

Examples of point plotted pictures:



Though no continuous lines are drawn but only a sense of points are being made bright, because of the properties of the human eye, we see continuous lines, when the points that are being lighted are fairly close to each other.

In fact, the closer the points to one another, we see better pictures (see the example below)

In the above figure both pictures indicate A, but in the second picture, the points are closer and hence it appears more like A than the first. How many points are there per unit area of the screen indicate what is known as the "resolution" of the monitor. Higher the resolution, we get more number of points and hence better quality pictures can be displayed (As a corollary, such high resolution monitors are costlier)

Incremental methods

The concept of incremental methods, as the name suggests, is to draw the picture in stages - incrementally. I.e. from the first point of the picture, we have a method of drawing the second point, from there to the third point etc. They are also sometimes called "iterative methods" because they draw picture in stages - in iterations.

Qualities of good line drawing algorithms: Before we start looking at a few basic line drawing algorithms, we see what are the conditions that they should satisfy. While the same picture can be drawn using several algorithms, some are more desirable than others, because they provide as features that enable us to draw better "quality" pictures. A few of the commonly expected qualities are as follows:

i. **Lines should appear straight:** Often straight lines drawn by the point plotting algorithms do not appear all that straight.

Incremental methods

The concept of incremental methods, as the name suggests, is to draw the picture in stages - incrementally. I.e. from the first point of the picture, we have a method of drawing the second point, from there to the third point etc. They are also sometimes called "iterative methods" because they draw picture in stages - in iterations.

Qualities of good line drawing algorithms: Before we start looking at a few basic line drawing algorithms, we see what are the conditions that they should satisfy. While the same picture can be drawn using several algorithms, some are more desirable than others, because they provide as features that enable us to draw better "quality" pictures. A few of the commonly expected qualities are as follows:

i. **Lines should appear straight:** Often straight lines drawn by the point plotting algorithms do not appear all that straight.

Generation of Circles

The above algorithms can always be extended to other curves - the only required condition is that we should know the equations of the curve concerned in a differential form. We see a few cases.

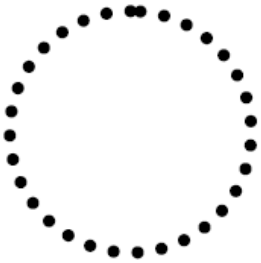
i) A circle generating DDA:

The differential equation of a circle is $\frac{dy}{dx} = -x/y$

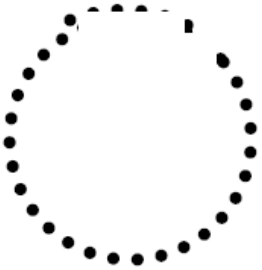
Hence by using the above principle, we can implement the circle plotting DDA by using the following set of equations $x_{n+1} = x_n + \varepsilon y_n$ and $y_{n+1} = y_n - \varepsilon x_n$

Where the subscript n refers to the present value and n+1 to the next value to be computed. Δy and Δx are the increments along the x and y values.

Unfortunately, this method ends up drawing a spiral instead of a circle, because the two ends of a circle do not meet. This is because, at each stage, we move slightly in a direction perpendicular to the radius, instead of strictly along the radius i.e. we keep moving slightly away from the center. So, in the end, we get the closing point a little higher up than where it is required and hence the circle does not close up



Ideal Circle



Drawn by a DDA

```
/* Program to demonstrate circle using DDA algorithm */
    # include <graphics.h>
    # include<conio.h>
    # include<dos.h>
    #include<alloc.h>
    #include<math.h>

    void main()
    {
    int gm,gd=DETECT,I,;
    int x,y,x1,y1,j;
    initgraph(&gd,&gm,"");
    x=40;  /*The c0-ordinate values for calculating radius */
    y=40;
    for(i=0;i<=360;i+=10)

{
          setcolor(i+1);
          x1=x*cos(i*3.142/180)+y*sin(i*3.142/180);
   y1=x*sin(i*3.142/180)-y*cos(I*3.142/180);
   circle(x1+getmaxx()/2,y1+getmaxy()/2,5); /* center of the circle is center
   of the screen*/
      delay(10);
}
getch();
}
```

The following program draws the circle using Bresenham's algorithm.

```
/* program to implement Bresenham's Circle Drawing Algorithm */

  # include<stdio.h>
# include<conio.h>
# include <graphics.h>
# include<math.h>
#include<dos.h>

/* Function for plotting the co-ordinates at four different angles that are placed
at egual distences */
```

```
void plotpoints(int xcentre, int ycentre,int x,int y)
{
int color=5;
putpixel(xcentre+x,ycevtre+y,color);
putpixel(xcentre+x,ycevtre-y,color);
putpixel(xcentre-x,ycevtre+y,color);
putpixel(xcentre-x,ycevtre-y,color);


putpixel(xcentre+y,ycevtre+x,color);
putpixel(xcentre+y,ycevtre-x,color);
putpixel(xcentre-x,ycevtre+x,color);
putpixel(xcentre-y,ycevtre-x,color);
}
```

```
/* Function for calculating the new points for(x,y)co-ordinates. */

void cir(int xcentre, ycentre, int radius)
{
int x,y,p;
x=0; y=radius;
plotpoints(xcentre,ycentre,x,y);
p=1-radius;
while(x<y)
{
if(p<0)
p=p+2*x+1:
else
```

```
{
  y--;
p=p+2*(x-y)+1;
}
x++;
plotpoints xcentre, ycentre,x,y);
delay(100);
}
}
```

/* The main function that takes (x,y) and 'r' the radius from keyboard and activates other functions for drawing the circle */

```
void main()

{
intgd=DETECT,gm,xcentre=200,ycentre=150,redius=5;
printf("\n enter the center points and radius  :\n");
scanf("%d%d%d", &xcentre, &ycentre, &radius);
clrscr();
initgraph(&gd,&gm,"");
putpixel(xcentre,ycentre,5);
cir(xcentre,ycentre,redius);
getch();
closegraph();
}
```

Bresenham specified the algorithm for drawing the ellipse using mid point method. This is illustrated in the following program.

/* BBRESENHAM's MIDPOINT ELLIPSE ALGOTITHM. */

```
  # include<stdio.h>
# include<conio.h>
# include<math.h>

# include <graphics.h>
int xcentre, ycentre, rx, ry;
int p,px,py,x,y,rx2,ry2,tworx2,twory2;
void drawelipse();
void main()
{
```

```
int gd=3,gm=1;
clscr();
initgraph(&gd,&gm,"");
printf("n Enter X center value: ");
scanf("%d",&xcentre);
printf("n Enter Y center value: ");
scanf("%d",&ycentre);
printf("n Enter X redius  value: ");
scanf("%d",&rx);
printf("n Enter Y redius  value: ");
scanf("%d",&ry);
cleardevice();
ry2=ry*ry;
rx2=rx*rx;
twory2=2*ry2;
tworx2=2*rx2;

 /* REGION first */
 x=0;
 y=ry;
 drawelipse();

 p=(ry2-rx2*ry+(0.25*rx2));
 px=0;

 py=tworx2*y;
 while(px<py)
 {
 x++;
 px=px+twory2;
 if(p>=0)
 {

       y=y-1;
       py=py-tworx2;
  }
```

```
if(p<0)
    p=p+ry2+px;
else
      {

 p=p+ry2+px-py;


     drawelipse();
      }
 }

 /*REGION  second*/
   p=(ry2*(x+0.5)*(x+0.5)+rx2*(y-1)*(y-1)-rx2*ry2);
    while(y>0)
    {
     y=y-1:
      py=py-tworx2;
      if(p<=0)
      {
       x++;
       px =px + twory2;
 }
 if(p >0)
      p=p+rx2-py;
 else
 {
     p=p+rx2-py+px;
     drawelipse();
     }
 }
  getch();
  closegraph();
 }
 void drawelipse()
 {
 Putpixel (xcenter +x, ycenter +y, BROWN);
 putpixel (xcenter +x, ycenter +y, BROWN);
 putpixel (xcenter +x, ycenter +y, BROWN);
 putpixel (xcenter +x, ycenter +y, BROWN);
 }
```

TWO DIMENSIONAL TRANSFORMATIONS

Introduction

In this unit, you are introduced to the basics of pictures transformations. Graphics is as much about the concept of creating pictures as also about making modifications in them. Most often, it is not changing the pictures altogether, but about making "transformation" in them. Like shifting the same picture to some other place on the screen, or increasing or decreasing it's size (this can be in one or two directions) or rotating the picture at various angles - The rotation also can be either w.r.t. the original x, y coordinates or with any other axis. All these are essentially mathematical operations. We view points (and hence pictures, which are nothing but the collections of points) as matrices and try to transform them by doing mathematical operations on them. These operations yield the new pixel values, which, when displayed on the CRT give the transformed picture.

What is transformation?

In the previous unit, we have seen the concept of producing pictures, given their equations. Though we talked of generating only straight lines and circles, needless to say similar procedures can be adopted for the other more complex figures - in many cases a complex picture can always be treated as a combination of straight line, circles, ellipse etc., and if we are able to generate these basic figures, we can also generate combinations of them. Once we have drawn these pictures, the need arises to transform these pictures. We are not essentially modifying the pictures, but a picture in the center of the screen needs to be shifted to the top left hand corner, say, or a picture needs to be increased to twice it's size or a picture is to be turned through 90^0. In all these cases, it is possible to view the new picture as really a new one and use algorithms to draw them, but a better method is, given their present form, try to get their new counter parts by operating on the existing data. This concept is called transformation.

Matrix representation of points

Before we start discussing about the actual transformations, we would go through the concept of representation of points. Once we know how to unambiguously represent a pant, we will be able to represent all other possible pictures.

Normally, we represent a point by two values in a rectangular coordinate systems as (x,y). x represents the distance of the point from the origin in the horizontal direction and y in the vertical directions. Negative values are intended to represent movement in the reverse direction (on a CRT screen, however, negative valued pixels can not be represented).

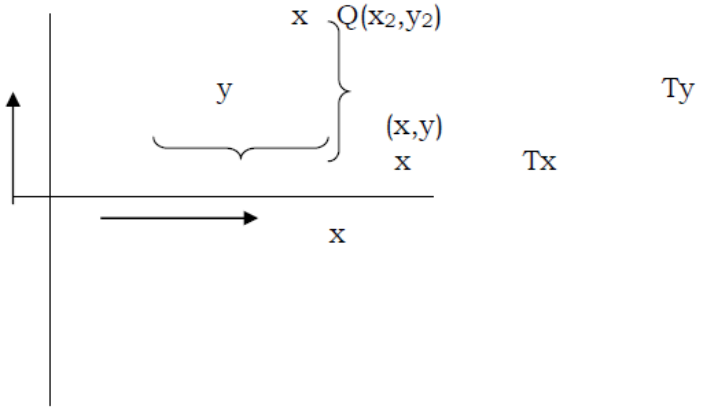
However, in the context of graphics we tend to represent a point as a 3 valued entity [x y 1] where x and y are the coordinates and 1 is just added to the representation. But use of this additional value becomes significant shortly.

The basic Transformation

Now we are ready to probe into the basics of transformations. As indicated earlier, we talk about transforming points, throughout the discussions, but any complex picture can be transferred using similar techniques in succession.

The three basic transformations are (i) Translation (ii) rotation and (iii) scaling. Translation refers to the shifting of a point to some other place, whose distance with regard to the present point is known. Rotation as the name suggests is to rotate a point about an axis. The axis can be any of the coordinates or simply any other specified line also. Scaling is the concept of increasing (or decreasing) the size of a picture. (in one or in either directions. When it is done in both directions, the increase or decrease in both directions need not be same) To change the size of the picture, we increase or decrease the distance between the end points of the picture and also change the intermediate points are per requirements;

Translation



Consider a point P(x_1, y_1) to be translated to another point Q(x_2, y_2). If we know the point value (x_2, y_2) we can directly shift to Q by displaying the pixel (x_2, y_2). On the other hand, suppose we only know that we want to shift by a distance of Tx along x axis and Ty along Y axis. Then obviously the coordinates can be derived by $x_2 = x_1 + T_x$ and $Y_2 = y_1 + T_y$.

Rotation

Suppose we want to rotate a point (x_1 y_1) clockwise through an angle/ about the origin of the coordinate system. Then mathematically we can show that

$$\mathbf{x_2 = x_1\cos\theta + y_1\sin\theta \quad and}$$
$$\mathbf{y_2 = x_1\sin\theta - y_1\cos\theta}$$

These equations become applicable only if the rotation is about the origin.

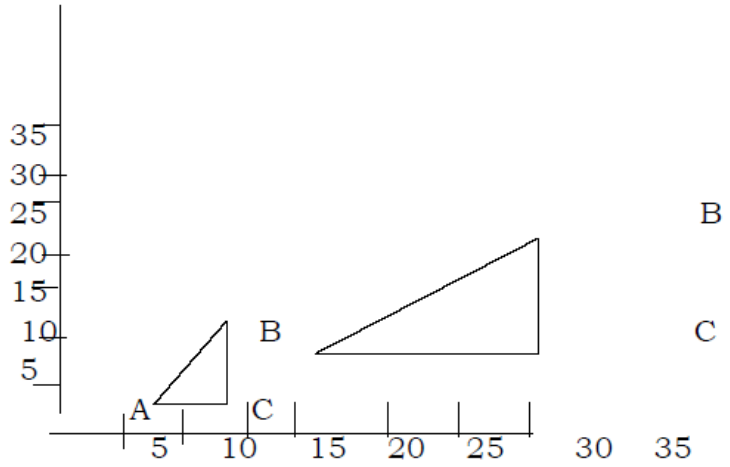
In the matrix for $[x_2\ y_2\ 1] = [x_1\ y_1\ 1] * \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Scaling

Suppose we want the point (x_1 y_1) to be scaled by a factor s_x and by a factor s_y along y direction.

Then the new coordinates become: $x_2 = x_1 * s_x$ and $y2 = y_1 * s_y$

(Note that scaling a point physically means shifting a point away. It does not magnify the point. But when a picture is scaled, each of the points is scaled differently and hence the dimension of the picture changes.)



For example consider a Triangle formed by the points A (5,5), B(10,10) and C (10,5). Suppose we scale it by a factor of 3 along x-axis and 2 along y-axis.

Then the new points will A(5 * 3, 5 * 2)
B(10*3, 10*2) and
C(10*3,5*2)

In the matrix form we get

$$[x_2 \quad y_1 \quad 1] = [x_1 \; y_1 \quad 1] * \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

THREE DIMENSIONAL TRANSFORMATIONS

Introduction
Three Dimensional transformation
Translations
Scaling
Rotation

Introduction

In this unit, we look into the basics of 3-D graphics, beginning with transformations. In fact the ability to transform a 3-dimensional point, i.e. a point represented by 3 Co-ordinates (x,y,z) is of immense importance not only for the various operations on the picture, but also for the ability to display the 3-D picture in a 2-D screen. We briefly see the various transformation operations – they are nearly similar to the 2-D operations. We also see the concepts of clipping and windowing in 3-D.

Three Dimensional Transformation

Just as in the case of 2D, we represent the transformation operations as a series of matrix operations. With this, we obtain the flexibility of sequencing a series of operations one after the other to get the desired results on one hand and also the ability to undo the operations, by resorting to the reverse sequence. Since in the 2-dimensional case we were representing a point (x,y) as a tuple [x y 1], in the 3-dimensional case, we represent a point (x,y,z) as a [x y z 1]. The dimensions of the matrices grow from 3 x 3 to 4 x 4.

Translations

Without repeating the earlier methods, we simply write

$$[x_1 \;\; y_1 \;\; z_1 \;\; 1] = [x \;\; y \;\; z \;\; 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{pmatrix}$$

Where the point [x y z 1] gets transformed to [x_1 y_1 z_1 1] after translating by Tx, Ty and Tz along the x,y,z directions respectively.

Scaling

A given point [x y z 1] gets transformed to [x_1 y_1 z_1 1] after getting scaled by factors Sx, Sy and Sz in the three dimensions to

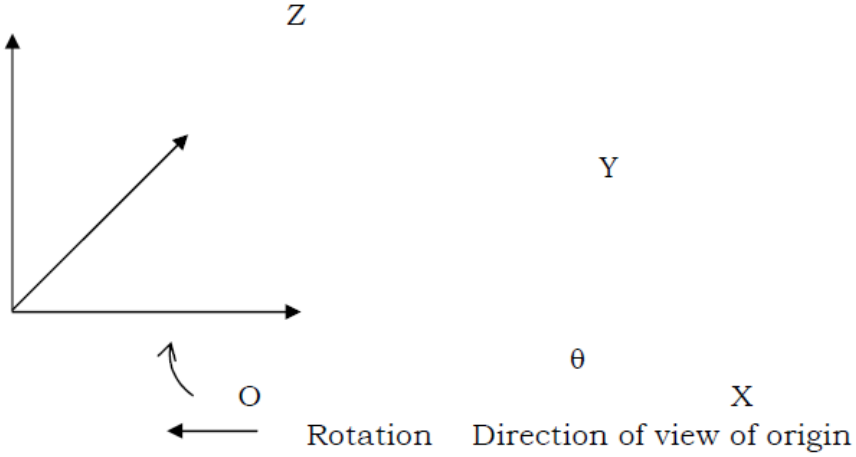
$$[x_1 \;\; y_1 \;\; z_1 \;\; 1] = [x \;\; y \;\; z \;\; 1] \begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

Rotation in 3-dimensions is a more complex affair. (In fact, even in 2 dimensions, rotation was more involved than scaling or translation because the concept of point of rotation). This is because; the rotation takes place about an axis. The same point, given the same amount of rotation, gets transformed to different points depending on which axis it was rotated.

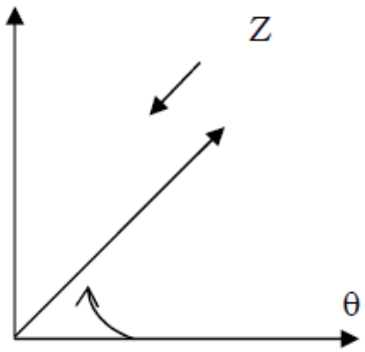
The simplest of the cases is to rotate the point about an axis that passes through the origin, and coincides with one of the axes x, y or z. The next complication arises when the axis passes through the origin, but does not coincide with any of the axes. The most general case would be, of course, when an arbitrary axis that does not pass through the origin becomes the axis of rotation.

Let us begin with simplest cases: The understanding is that a clockwise rotation, when viewed at the origin, standing on the axis is taken as positive and the other direction is negative. If this description looks too complicated, look at the following figures. In each case, we write down the transformation for the rotation through a positive angle of +θ.



$$[x_1 \; y_1 \; z_1 \; 1] = [x \; y \; z \; 1] \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 1 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation Matrix



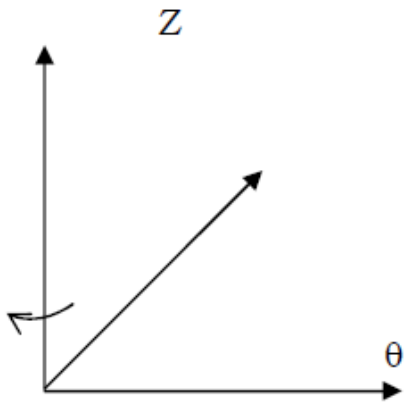
Direction of view of
Origin

Y

O X

$$[x_1 \ y_1 \ z_1 \ 1] = [x \ y \ z \ 1] \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation Matrix



Direction of view of origin

Y

O X

$$[x_1 \ y_1 \ z_1 \ 1] = [x \ y \ z \ 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation Matrix

HIDDEN SURFACE REMOVAL

Introduction
Need for hidden surface removal
The Depth - Buffer Algorithm
Warnock's Algorithm

Introduction

In this unit, you will be introduced to one of the most interesting and involved concept of computer graphics – the concept of hidden surface elimination. When two or more object are represented one behind the other – it is quite clear that some of them either partially or fully obscure the other object in such cases the hidden parts of the objects are to be removed.

Several algorithms for the same are introduced. Almost all of them work on the simple concept of sorting the polygons in the order of their distance, the nearest ones being represented in full, the farther ones in part since, in raster graphics, a given pixel can represent more than one object (each with same x,y) it will represent that object that is nearest to the screen amongst these objects. Though this concept is straight forward and accurate, it suffers from the difficulty that it is computationally intensive. Hence, several efficient algorithms, which perform the same job with more efficiency, are introduced. Most of them make use of some sort of coherence concept – i.e. pixels in the neighborhood of a pixel share the properties of a pixel. i.e. If a pixel forms a part of an object, the neighboring pixel also, most probably, form the part of the same object. You will also be introduced to certain specific instances, wherein these concepts may not yield satisfactory results.

Need for hidden surface removal

This has been considered as one of the most challenging jobs of computer graphics. Once we start talking of solid objects in 3 dimensional spaces, it is implied that some of the objects that are nearer to the viewer tend to partly or wholly cover other objects. In fact, even if there is only one object, some of its faces are unseen (the back faces) and some are partially seen (the side faces). The ability to identify the faces and surfaces that are to be covered and the extent of coverage in the case of partially covered surfaces in real time is not only computationally intensive, but also analytically daunting. When only wire frame types of drawings are being displayed, the task gets somewhat simplified to that of "hidden line removal" – identifying those lines that should not be shown. However, when solid objects are being considered, the task becomes more complex because entire surfaces need to be identified for removal.

A large number of algorithms are available for the job –though no single algorithm can be though to be all encompassing capable of being efficient in all possible conditions. However almost all of them share some common feature. The first one is that at some point in the algorithm, they tend to sort the objects in the order of their Z-distance from the viewer and try to eliminate the farthest ones. But the sorting tends to be a difficult task at least in some cases, since often an object may not be identified with a unique distance – Z. when several part of the object have different Z coordinates, simple, direct sorting methods may become inadequate.

The other common feature with these algorithms in the use of coherence. As we have seen in other contexts, the coherence (or similarity with respect to a property) between neighboring pixels is used to reduce the number of computations effectively.

The behavior of the algorithm also depends on which type of images one is talking of. In the case of line drawing algorithms, the problem is solved using the various properties of lines, whereas in the case of raster images, the algorithms tend to look like extensions of 2-dimensional scan conversion algorithms.

The algorithms can also work either with respect to the object space or the image space. One should clearly be able to draw the distinguishing line between them. The object space in the space occupied by the pictures created by the algorithms. However, before these pictures can be displayed, they undergo various operations – like clipping, windowing, perspective transformations etc. This final set of pictures – ready for display on the screen is called the image space. The object space algorithms tend to calculate the values with as a precision as feasible since often these calculations form the basis for the next set of calculations, whereas the image space algorithms calculate with precision that is in line with the precision available with the display devices. This is because any higher precision, achieved with great efforts, will become useless since the display devices cannot anyway handle such precisions. Further, the computational efforts in the case of objects –

since every object tend to rapidly increase with the no. of objects – since every object will have to be tested with other objects, where as in the image apace computations, the increase is much slower, since one tends to look at the number of pixels, irrespective of the no. of objects in the scene. The number of pixels in a given resolution of display device is a constant.

Having noted some of the expected features of the algorithms, we now look into the working of some of the algorithms.

The Depth – Buffer algorithm

The concept of this algorithm is extremely simple and straightforward. Given a given resolution of the screen, every pixel on the screen can represent a point on one (and only one) object (or it may be set to the back ground if it does not form a part of any object). I.e. irrespective of the number of objects in line with the pixel, it should represent the object nearest to the viewer. The algorithm aims at deciding for every pixel the object whose features it should represent. The depth-buffer algorithm used two arrays, depth and intensity value. The size of the arrays equals the number of pixels. As can be expected, the corresponding elements of the array store the depth and intensity represented by each of the pixels.

a. For every pixel, set it's depth and intensity pixels to the back ground value ie. At the end of the algorithm, if the pixel does not become a part of any of the objects it represents the background value.

b. For each polygon on the scene, find out the pixels that lie within this polygon (which is nothing but the set of pixels that are chosen if this polygon is to be displayed completely).

For each of the pixels

i) Calculate the depth Z of the polygon at that point (note that a polygon, which is inclined to the plane of the screen will have different depths at different points)

ii) If this Z is less than the previously stored value of depth in this pixel, it means the new polygon is closer than the earlier polygon which the pixel was representing and hence the new value of Z should be stored in it. (i.e from now on it represents the new polygon). The corresponding intensity is stored in intensity vector.

If the new Z is greater than the previously stored vale, the new polygon is at a farther distance than the earlier one and no changes need be made. The polygon continues to represents the previous polygon.

One may note that at the end of the processing of all the polygons, every pixel, will have the intensity value of the object which it should display in its intensity location and this can be displayed.

This simple algorithm, as can be expected, works on the image space. The scene should have properly projected and clipped before the algorithm is used.

The basic limitation of the algorithm is it's computational intensiveness. On a 1024 X 1024 screen it will have to evaluate the status of each of these pixels in a limiting case. In it's present form, it does not use any of the coherence or other geometric properties to reduce the computational efforts.

To reduce the storage, some times the screen is divided into smaller regions like say 50 X 50 or 100 X 100 pixels, computations made for each of this religions, displayed on the screen and then the next region is undertaken. However this can be both advantageous and disadvantageous. It is obvious that such a division of screen would need each of the polygons to be processed for each of the regions – thereby increasing the computational efforts. This is disadvantage. But when smaller regions are being considered, it is possible to make use of various coherence tests, thereby reducing the number of pixels to be handled explicitly.

Warnock's Algorithm

This is one of the class of "area" algorithms. It tries to solve the hidden surface problem recursively. The algorithm proceeds on the following lines.

i. Try to solve the problem by taking the entire screen as one window. If no polygons overlap either in x or y or even if they do, overlap so that they do not obscure, then return the screen.

ii. If the problem is not easily solvable in step (i) the algorithm divides the screen into 4 equal parts and tries to apply step (i) each of them. If it is not solvable, again divides into smaller windows and so on.

iii. The recursive process continues till each window is trivially solvable or one endsup with single pixels.

We have still not described how the actual "solution" is done. To do this, in any window, the algorithm classifies the polygons into three groups

i) Disjoint Polygons: Polygons that do not overlap in the window and hence can be trivially passed.

ii) A bigger and a smaller polygon overlapping so that the smaller one will be completely blocked by the bigger one (if the Z of the larger polygon is smaller than Z of the smaller one).

iii) Intersected polygons: Polygons that partly obscure each other.

Polygons that fall into category (i) and (ii) are removed at each level. If the remaining polygons can be easily solved, the recursive process stops at that level, else the process continues (with the polygons of category (i) and (ii) removed).

Since at each recursive level a few polygons are removed, as the windows become smaller and smaller with the advance of recursion, the list of polygons falling into them also reduces and hopefully the problem of hidden surfaces gets solved trivially.

One main draw back of algorithm is that the windows get divided into smaller and smaller rectangles. In many cases it would be efficient if one can divide the window roughly in the shape of the polygons themselves. Such an algorithm, developed by Wieler and Atherton, was found more efficient, though more complex in terms of larger complexities of recursive divisions and clippings.

UNIT-II

Definition of CAD Tools, Types of system

CAD or Computer Aided Design software was introduced in the late 1960's to expedite engineering drawing process.

While CAD is used mainly in engineering drawing and construction architecture, it can also used for other purposes.
There are various flavours of CAD available today and there are different methods of classifying them.
Types of CAD Software

2 Dimensional CAD (2D CAD)
2D CAD is the pioneer of CAD software, and was developed in the early 70s. At that time, major automobile, aerospace and other engineering companies developed in-house tools to automate repetitive drafting requirements. 2D CAD relies on basic geometric shapes like lines, rectangles, circles, etc. to produce flat drawings.

These types of softwares have been first developed way back in 1970's. AutoDesk is one of the pioneering companies that has played a significant role in developing CAD software.

3 Dimensional CAD (3D CAD)
3D CAD is a step up from the 2D CAD software of yesteryears. As the processing power of computers increased and the graphic display capabilities improved, 3D CAD has become an increasingly popular design tool. 3D CAD allows creation of 3D images that are realistic. These images are called 3D models as they can be viewed and rotated in any direction – X, Y or Z. You can also display views from a 3D model, such as isometrics or perspectives, from any angle using 3D CAD. 3D CAD tools were introduced in 1980's by a partnership between IBM-Dassults. 3D CAD quickly became popular because of enhanced visual capability.
The rapid advancement of 3D software today has helped quick turnaround in product design, giving birth to the concept for product lifecycle management (PLM). A few of today's leading 3D CAD software includes SolidEdge and SolidWorks. Of course, with the vast array of tools, professional training is needed to master these tools.

There is yet another way of classifying CAD software - in terms of their operating parameters. Once youunderstand these parameters, you can optimize the CAD software properly. A little training should help you go a long way!
Single-file-mode systems - This type of CAD software allows only a single user to work on a single file at a time.

Referenced-file-mode systems - In this type of CAD software, users can work on their own files with the files of other users attached as a background. This enables users to levergae other users' work as background data.
Collaborative-mode systems - These CAD systems take the referenced-mode system to the next level. They allow a team of users to collaboratively work with each other's data and see the changes other users make to the data as they go. And of course, the giants in this field (for example AutoCAD) can be used in different modes of a operation.
3D CAD can be further classified as:
Wire-frame models – they create skeleton like models with lines and arcs. Since they appear to be made of wires, and everything in the background is visible, they are called wire-frame models. They are not very popular anymore.
Surface models – unlike wire frames, these models are created by joining 3D surfaces. Since nothing in the background is visible, the surface models are quite realistic.
Solid models – they are considered to be the most useful CAD models. Although they appear to be the same as surface models, they also have additional properties like weight, volume and density, just like actual physical objects. These models are commonly used as prototypes to study engineering designs.

CAD/CAM system evaluation criteria

1. CAPABLE, EFFICIENT 3D DESIGN

The centerpiece of 3D CAD is a 3D master model that's used for all aspects of manufacturing: product design and simulation, drafting, tool design, numerically controlled tool programming, and inspection. The 3D model must accurately represent every part in your company's products and the relationships among them. To maximize efficiency, designers should be able to design in 3D with as few steps as possible without compromising design quality.

When evaluating CAD software, find out how efficient each package is at creating the types of products your company makes. For example, if your company makes sheet metal parts, pay attention to the special aids for modeling them and automatically generating flat patterns. If your firm designs stylish products, look at the tools for creating freeform surfaces and blends with continuous curvature. Designers of machinery should examine how easily they can assemble large numbers of parts and insert purchased parts, such as fasteners and electrical components, from a library. Because changes are inevitable, assess how hard it is to modify parts and assemblies.
A CAD system that can make your company's designs with even 20 percent fewer steps will offer important cost advantages compared with systems that are less efficient. A 3D CAD system that offers the best value will combine exceptional technical capabilities with reasonable cost-of-ownership.

2. INFORMATION FLOW THROUGH EXTENDED ENTERPRISES

In today's world, few manufacturers are vertically integrated. Most rely on global communities of suppliers for parts, tools, subsystems, production equipment, and design. Whether your company is a supplier, a customer, or both, it can benefit from sharing 3D CAD models with others. When possible, choose a CAD system that's popular in your industry and supplier community. This choice helps eliminate the need to translate files from one system to another. Translation takes time and can introduce errors.

Also look at each system's ability to import files from other systems. Make sure your CAD system supports international standards such as STEP, IGES, VDA, and IDF. Evaluate the tools for fixing damage to imported shapes. How easy are they to use? How well do they work? If your firm must translate many files from several brands of CAD systems, check out the direct translators available with each CAD system and also those from third parties that specialize in translation software. Don't limit your evaluation of data sharing to file exchanges. Systems based on shared internet hosts enable designers to collaborate in real time with customers to explore options and identify good solutions quickly. Sharing CAD data with customers or suppliers can save thousands of hours and weeks of schedule time compared with the cost of remastering them interactively. The ability to collaborate in 3D on products and processes can reduce costs while helping to deliver better products.

3. DRAFTING TOOLS THAT MEET YOUR STANDARDS TODAY AND INTHE FUTURE

Even though you'll be designing in 3D, your suppliers and factory workers may need drawings. A clear drawing shows information that isn't obvious in a 3D model: critical dimensions and tolerances, material and surface-finish specifications, and notes about processing, such as curing or heat treatment. Be sure any 3D CAD system you buy can make drawings to your current standards for dimensions, tolerances, lettering, and parts lists. And be sure your drawings can be exported in popular formats, such as PDF, DXF, and DWG. But 3D CAD is changing drawings as we know them. Leading manufacturers are employing annotated 3D models that convey the information found on drawings without a separate document.

This so-called “model-based definition” saves drafting time, simplifies product-data management, and enables automated manufacturing and inspection systems to read dimensions and tolerances directly from 3D models, helping to eliminate errors.

4. TOOLS TO TAKE YOUR DESIGNS FROM CONCEPT THROUGH MANUFACTURING

Designs don’t make money until physical products are delivered. Look for 3D CAD software with a rich variety of applications that can reduce not only design time, but testing, machining, cost estimating, and inspection. Companies that design systems to order can benefit from software that generates parts and assemblies automatically in response to customer specifications. Such tools may be general purpose, such as configuration software, or special tools optimized for designing products such as mold assemblies or stamping dies.

Because physical testing is costly and slow, you should look to reduce the number of physical tests by simulating physical behavior, such as kinematics, dynamics, stress, deflection, vibration, temperatures, or fluid flow. Look for a system that has integrated analytical tools or efficient interfaces to your preferred simulation software. Software for designing electrical wiring can help reduce errors and ensure machinery is wired correctly. Cost-estimating software enables designers to hit cost targets by revising designs sooner instead of waiting for estimators to say they are over budget. Inspection software can slash the time needed to prepare documents for inspecting parts on delivery.

Picking the right add-in applications for your company’s business can slash the time needed to bring products to market. To make sure you have the best tools, choose a CAD platform that gives a broad choice of solutions. It should have an extensive and well-documented application programming interface (API). Good APIs make it less costly for third parties to integrate specialty applications with your CAD system. And they let your own programmers write software tailored to your ways of using CAD models.

5. HELP MANAGING DATA

Organizations with more than just a few designers can benefit from product data management (PDM) software integrated with their CAD tools. Because relationships among files in 3D

systems are so complex, an automated system to store and organize them is essential. Without PDM, designers can unknowingly overwrite each other's work, reinvent parts that have already been designed, and send the wrong revision levels to manufacturers. Together, these sorts of errors can waste hundreds of hours of work each year and thousands of dollars in defective parts.

PDM systems do much more than store and organize files. They also help designers find existing parts to re-use instead of reinventing them, generate materials lists for cost estimating, and feed data to manufacturing resource planning (MRP) systems. More advanced PDM software can automate change-control processes to ensure that out-of-date or unreleased information isn't sent to factories or suppliers.

6. INNOVATIVE R&D TO PROTECT YOUR INVESTMENT

Computing technology is constantly changing. If your CAD vendor doesn't take advantage of this evolution, in a few years you'll find that your organization has an obsolete and costly-to-maintain CAD system. Buy from suppliers that have a proven record of being manufacturing industry leaders with large and sophisticated R&D teams.

7. PLEASANT BUSINESS RELATIONSHIPS

Believe it or not, some of the greatest sources of friction between buyers of CAD software and their customers are the nontechnical business aspects of the relationship. Just as some airlines annoy customers with extra fees for checked baggage, flight changes, drinks, and blankets, some CAD suppliers levy hidden charges for software and services that most customers need.

To avoid aggravation and lower your costs, look for suppliers who offer straightforward software packages that have what you need. Look at the terms for floating licenses that enable designers who don't need CAD full-time to share licenses. And be sure your best designers can use the software both at work and at home without hassles.

8. SHORT LEARNING CURVE

Adopting 3D methods requires training and experience. So choose a system that's easy to learn as well as capable. Look for a system that has a consistent user interface throughout. Be sure design and manufacturing procedures flow logically from start to finish. Some systems have hidden dungeons and dragons that stop designers halfway through a task and make them start over.

Developing your own training materials is costly. Choose a system with built-in tutorials, a rich array of computer-based training aids, and a vibrant online community that lets workers ask questions and get answers. You'll also want a system that's taught in local schools and universities so you can hire students who are ready to work.

9. WHO CAN HELP YOU

A successful relationship with your CAD software dealer only begins with the sale. Buy from a dealer with the skills and experience to help you successfully integrate 3D design with manufacturing. Find out how many 3D customers potential dealers have trained and supported. Look at the availability of quality training classes. Does the dealer support a viable user group? Does the reseller offer ongoing training classes to help you improve your design and manufacturing processes?

Ask for the resumes of the technical staff and interview them before you buy. Ask reference customers if the dealer's technical staff is capable of solving tough problems. Good local support can make the difference between a costly adoption of new CAD software and one that advances your business objectives now and in the future.

CAD/CAM Systems Evaluation Criteria

The various types of CAD/CAM systems are Mainframe-Based Systems, Minicomputer-Based Systems, Microcomputer-Based Systems and Workstation-Based Systems.

The implementation of these types by various vendors, software developers and hardware manufacturers result in a wide variety of systems, thus making the selection process of one rather difficult. CAD/CAM selection committees find themselves developing long lists of guidelines to screen available choices.

These lists typically begin with cost criteria and end with sample models or benchmarks chosen to test system performance and capabilities. In between comes other factors such as compatibility requirements with in-house existing computers, prospective departments that plan to use the systems and credibility of CAD/CAM systems' suppliers.

In contrast to many selection guidelines that may vary sharply from one organization to another, the technical evaluation criteria are largely the same. They are usually based on and are limited by the existing CAD/CAM theory and technology. These criteria can be listed as follows.

System Considerations

(i) Hardware

- Each workstation is connected to a central computer, called the server, which has enough large disk and memory to store users' files and applications programs as well as executing these programs.

(ii) Software

- Three major contributing factors are the type of operating system the software runs under, the type of user interface (syntax) and the quality of documentation.

(iii) Maintenance

- Repair of hardware components and software updates comprise the majority of typical maintenance contracts. The annual cost of these contracts is substantial (about 5 to 10 percent of the initial system cost) and should be considered in deciding on the cost of a system in addition to the initial capital investment.

(iv) Vendor Support and Service

- Vendor support typically includes training, field services and technical support. Most vendors provide training courses, sometimes on-site if necessary.

Geometric Modeling Capabilities

(i) Representation Techniques

- The geometric modeling module of a CAD/CAM system is its heart. The applications module of the system is directly related to and limited by the various representations it supports. Wireframes, surfaces and solids are the three types of modeling available.

(ii) Coordinate Systems and Inputs

- In order to provide the designer with the proper flexibility to generate geometric models, various types of coordinate systems and coordinate inputs ought to be provided. Coordinate inputs can take the form of cartesian (x, y, z), cylindrical (r, θ, z) and spherical (θ, ϕ, z).

(iii) Modeling Entities

- The fact that a system supports a representation scheme is not enough. It is important to know the specific entities provided by the scheme. The ease to generate, verify and edit these entities should be considered during evaluation.

(iv) Geometric Editing and Manipulation

- It is essential to ensure that these geometric functions exist for the three types of representations. Editing functions include intersection, trimming and projection and

manipulations include translation, rotation, copy, mirror, offset, scaling and changing attributes.

(v) Graphics Standards Support

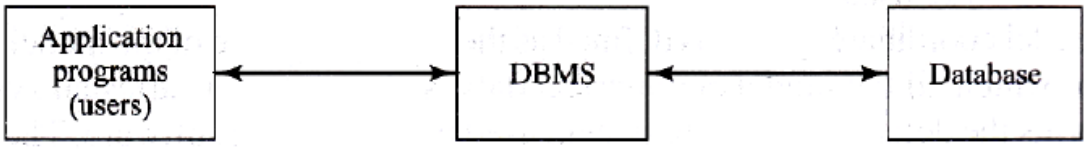
- If geometric models' databases are to be transferred from one system to another, both systems must support exchange standards.

CAD Softwares

- Softwares can be defined as an interpreter or translator which allows the user to perform specific type of application or job related to CAD.
- The user may utilize the software for drafting or designing of machine parts or components subjected to stresses or analysis of any type of system.
- The CAD application software can be prepared in variety of languages such as BASIC (Beginner's All-purpose Symbolic Instruction Code), FORTRAN, Java, PASCAL & C-Language.
- C- Language has been preferred for CAD software development because of no. of advantages as compared to other languages.
- The Java language has the capacity to operate with high level graphics and animation.

Database Management System (DBMS)

- A DBMS is defined as the software that allows access to use and/or modify data stored in a database. The DBMS forms a layer of software between the physical database itself (i.e., stored data) and the users of this database as shown in Fig. 1.31(a).
- DBMS shields users from having to deal with hardware-level details by interpreting their input commands and requests from the database. For example, a command such as retrieve a line could involve few lower-level steps to execute.



(*a*) Simplified DBMS

- In general, a DBMS is responsible for all database-related activities such as creating files, checking for illegal users of the database and synchronizing user access to the database.
- DBMSs designed for commercial business systems are too slow for CAD/ CAM. The handling of graphics data is an area where the conventional DBMSs tend to break down under the shear volume of data and the demand for quick display.
- By contrast, data handled in the commercial realm is mostly alphanumeric and the objects described are usually not very complex. A DBMS is directly related to the database model it is supposed to manage. For example, relational DBMSs require relatively large amounts of CPU time for searching and sorting data stored in the relations or tables.
- Therefore, the concept of database machines exists where a DBMS is implemented into hardware that can lie between the CPU of a computer and its database disks,

Graphics Exchange standards

- With the proliferation of computers and software in the market, it became necessary to standardize certain elements at each stage, so that investment made by companies in certain hardware or software was not totally lost and could be used without much modification on the newer and different systems.
- Standardization in engineering hardware is well known. Further, it is possible to obtain hardware and software from a number of vendors and then be integrated into a single system.
- This means that there should be compatibility between various software elements as also between the hardware and software. This is achieved by maintaining proper interface standards at various levels.

Both CAD/CAM vendors as well as users identified some needs to have some graphics standards. The needs are as follows:

i. **Software portability**: This avoids hardware dependence of the software. If the program is written originally for random scan display, when the display device is changed to raster scan display the program should work with minimum effort.
ii. **Image data portability**: Information and storage of images should be independent of different graphics devices.
iii. **Text data portability**: The text associated with graphics should be independent of different input/output devices.
iv. **Model database portability**: Transporting of design and manufacturing data from one application software to another should simple and economical.

The search for standards began in 1974 to fulfill the above needs both at the USA and International levels. As a result of worldwide efforts, various standards at different levels of the graphics systems were developed. The standards are as follows:

Both CAD/CAM vendors as well as users identified some needs to have some graphics standards. The needs are as follows:

i. **Software portability**: This avoids hardware dependence of the software. If the program is written originally for random scan display, when the display device is changed to raster scan display the program should work with minimum effort.
ii. **Image data portability**: Information and storage of images should be independent of different graphics devices.
iii. **Text data portability**: The text associated with graphics should be independent of different input/output devices.
iv. **Model database portability**: Transporting of design and manufacturing data from one application software to another should simple and economical.

The search for standards began in 1974 to fulfill the above needs both at the USA and International levels. As a result of worldwide efforts, various standards at different levels of the graphics systems were developed. The standards are as follows:

1. **GKS (Graphics kernel system):** It is an ANSI (American National Standards Institute and ISO (International Standards Organization) standard. It interfaces the application program with graphics support package.
2. **IGES (Initial graphics exchange specification):** It is an ANSI standard. It enables an exchange of model database among CAD/CAM software.
3. **PHIGS (Programmer's hierarchical interactive graphics system):** It supports workstations and their related CAD/CAM applications. It supports 3-dimensional modeling of geometry segmentation and dynamic display.
4. **CGM (Computer graphics metafile):** It defines functions needed to describe an image. Such description can be stored or transported from one graphics device to another.
5. **CGII (Computer graphics interface)**: It is designed to interface plotters to GKS or PHIGS. It is the lowest device independent interface in a graphics system.
6. **Drawing Exchange Format (DXF):** The DXF format has been developed and supported by Autodesk for use with the AutoCAD drawing files. It is not an industry standard developed by any standards organisation, but in view of the widespread use of AutoCAD made it a default standard for use of a variety of CAD/CAM vendors. A Drawing Interchange File is simply an ASCII text file with a file extension of .DXF and specially formatted text.
7. **Standard for the Exchange of Product Model Data (STEP)**, officially the ISO standard 10303, Product Data Representation and Exchange, is a series of International Standards with the goal of defining data across the full engineering and manufacturing life cycle. The ability to share data across applications, across vendor platforms and between contractors, suppliers and customers, is the main goal of this standard.

8. **Parasolid:** It is a portable "kernel" that can be used in multiple systems - both high-end and mid-range. By adopting Parasolid, start-up software companies have eliminated a major barrier to application development - a high initial investment. This enabled them to effectively market softwares with strong solid modeling functionality at lower-cost.
9. **PDES (Product Data Exchange Specification)** is an exchange for product data in support of industrial automation. "Product data" encompasses data relevant to the entire life cycle of a product such as design, manufacturing, quality assurance, testing and support. In order to support industrial automation, PDES files are fully interpretable by computer. For example, tolerance information would be carried in a form directly interpretable by a computer rather than a computerized text form which requires human intervention to interpret.

Curve Representation

Curve is defined as the locus of point moving with one degree freedom.

A curve can be represented by following two methods either by storing its analytical equation or by storing an array of co-ordinates of various points

Curves can be described mathematically by following methods:

(i) Non-parametric form
   a) Explicit form
   b) Implicit form
(ii) Parametric form

Non-parametric form:

- In this, the object is described by its co-ordinates with respect to current reference frame in use.

Explicit form: (Clearly expressed)

In this, the co-ordinates of y and z of a point on curve are expressed as two separate functions of x as independent variable.

$$P = [x \quad y \quad z]$$
$$= [x \quad f(x) \quad g(x)]$$

Implicit form: (Not clearly expressed)

In this, the co-ordinates of x, y and z of a point on curve are related together by two functions.

$F(x, y, z) = 0$

$G(x, y, z) = 0$

Limitation of nonparametric representation of curves are:

1. If the slope of a curve at a point is vertical or near vertical, its value becomes infinity or very large, a difficult condition to deal with both computationally and programming-wise. Other ill-defined mathematical conditions may result.
2. Shapes of most engineering objects are intrinsically independent of any coordinate system. What determines the shape of an object is the relationship between its data points themselves and not between these points and some arbitrary coordinate system.
3. If the curve is to be displayed as a series of points or straight line segments, the computations involved could be extensive.

Parametric form:

In this, a parameter is introduced and the co-ordinates of x, y and z are expressed as functions of this parameters. This parameter acts as a local co-ordinate for points on curve.

$$P(u) = [x \quad y \quad z]$$
$$= [x(u) \quad y(u) \quad z(u)]$$

Example 2.1: For the position vectors $P_1[1 \quad 2]$ and $P_2[4 \quad 3]$, determine the parametric representation of line segment between them. Also determine the slope and tangent vector of line segment.

A parametric representation of line is

$$P = P_1 + u(P_2 - P_1)$$

$$= [1 \quad 2] + u\,([4 \quad 3] - [1 \quad 2])$$

$$= [1 \quad 2] + u\,[3 \quad 1]$$

Parametric representation of x and y components are

$$x(u) = x_1 + u\,(x_2 - x_1)$$

$$= 1 + 3u$$

$$y(u) = y_1 + u\,(y_2 - y_1)$$

$$= 2 + u$$

The tangent vector is obtained by differentiating P(u)

$$P'(u) = [x'(u) \quad y'(u)]$$

$$= [3 \quad 1]$$

The slope of line segment is

$$\frac{dy}{dx} = \frac{dy/du}{dx/du}$$

$$= \frac{y'}{x'}$$

$$= \frac{1}{3}$$

Example 2.2 : The end points of line are P1(1,3,7) and P2(–4,5, –3). Determine

i. Tangent vector of the line
ii. Length of line
iii. Unit vector in the direction of line

Parametric representation of x, y and z components are

$$x(u) = x_1 + u\,(x_2 - x_1)$$
$$= 1 - 5u$$
$$y(u) = y_1 + u\,(y_2 - y_1)$$
$$= 3 + 2u$$
$$z(u) = z_1 + u\,(z_2 - z_1)$$
$$= 7 - 10u$$

Tangent vector, $P'(u) = [x'(u)\ \ y'(u)\ \ z'(u)]$
$= [-5\ \ 2\ \ -10]$
$= -5i + 2j - 10k$

Length of line, $L = |P_2 - P_1|$

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

$$= \sqrt{(-4-1)^2 + (5-3)^2 + (-3-7)^2}$$
$$= 11.358$$

Unit vector in the direction of line, $\hat{n} = \frac{P_2 - P_1}{|P_2 - P_1|} = \frac{P_2 - P_1}{L}$

$$= \frac{1}{11.358}[-5\ \ 2\ \ -10]$$
$$= [-0.44\ \ 0.176\ \ -0.88]$$
$$= -0.44i + 0.176j - 0.88k$$

Example 2.3: A line is represented by the end point $P_1(2, 4, 6)$ and $P_2(-3, 6, 9)$. If the value of parameter u at P_1 and P_2 is 0 and 1 respectively, determine the tangent vector for the line. Also determine the coordinate of a point represented by; u equal to 0, 0.25, -0.25, 1 and 1.5. Also find the length and unit vector of line between two points P_1 and P_2.

Parametric representation of x, y and z components are

$$x(u) = x_1 + u\,(x_2 - x_1)$$
$$= 2 - 5u$$
$$y(u) = y_1 + u\,(y_2 - y_1)$$
$$= 4 + 2u$$
$$z(u) = z_1 + u\,(z_2 - z_1)$$
$$= 6 - 3u$$

Tangent vector, $P'(u) = [x'(u)\ \ y'(u)\ \ z'(u)]$

$$= [-5\ \ 2\ \ -3]$$
$$= -5i + 2j - 3k$$

u	0	0.25	– 0.25	1	1.5
x (u)	2	0.75	3.25	–3	–5.5
y (u)	4	4.5	3.5	6	7
z (u)	6	5.25	6.75	3	1.5

Length of line, $L = |P_2 - P_1|$

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$
$$= \sqrt{(-3-2)^2 + (6-4)^2 + (9-6)^2}$$
$$= 6.16$$

Unit vector in the direction of line, $\hat{n} = \dfrac{P_2 - P_1}{|P_2 - P_1|} = \dfrac{P_2 - P_1}{L}$

$$= \frac{1}{6.16}[-5\ \ 2\ \ -3]$$
$$= [-0.81\ \ 0.324\ \ -0.487]$$
$$= -0.81i + 0.324j - 0.487k$$

Example 2.4: The two endpoints of diameter of a circle are $P_1(13,15,7)$ and $P_2(35,40,7)$. Determine the centre and radius of circle.

The centre of a circle, $P_c = \frac{1}{2}(P_1 + P_2)$

$$[x_c \;\; y_c \;\; z_c] = \left[\frac{x_1 + x_2}{2} \;\; \frac{y_1 + y_2}{2} \;\; \frac{z_1 + z_2}{2}\right]$$

$$[x_c \;\; y_c \;\; z_c] = \left[\frac{13+35}{2} \;\; \frac{15+40}{2} \;\; \frac{7+7}{2}\right]$$

$$[x_c \;\; y_c \;\; z_c] = [24 \;\; 27.5 \;\; 7]$$

The radius of circle

$$R = \frac{1}{2}\left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}\right)$$

$$R = \frac{1}{2}\left(\sqrt{(35-13)^2 + (40-15)^2 + (7-7)^2}\right)$$

$$R = 16.65$$

Parametric representation of synthetic curve

Analytic curves, are usually not sufficient to meet geometric design requirements of mechanical parts. Products such as car bodies, ship hulls, airplane fuselage and wings, propeller blades, shoe insoles and bottles are a few examples that require free-form, or synthetic, curves and surfaces.

The need for synthetic curves in design arises on two occasions: when a curve is represented by a collection of measured data points and when an existing curve must change to meet new design requirements.

In the latter occasion, the designer would need a curve representation that is directly related to the data points and is flexible enough to bend, twist, or change the curve shape by changing one or more data points.

Data points are usually called control points and the curve itself is called an interpolant if it passes through all the data points.

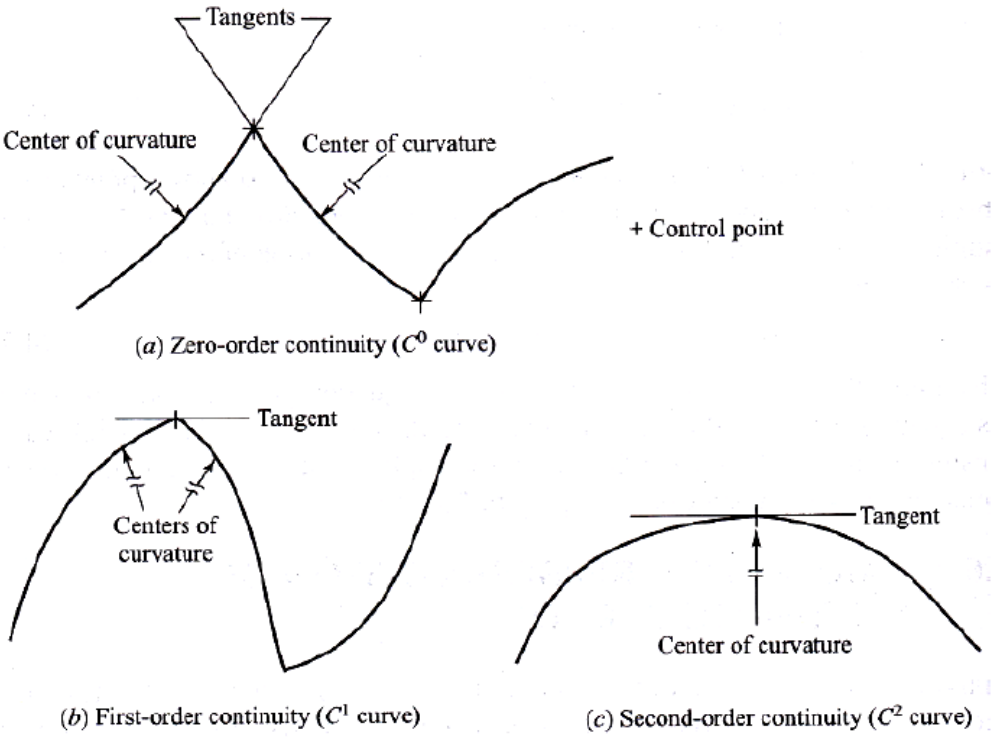


Fig. 2.11 Various Orders of Continuity of Curves

Mathematically, synthetic curves represent a curve-fitting problem to construct a smooth curve that passes through given data points. Therefore, polynomials are the typical form of these curves.

Various continuity requirements can be specified at the data points to impose various degrees of smoothness of the resulting curve. The order of continuity becomes important when a complex curve is modeled by several curve segments pieced together end to end.

Zero-order continuity (C^0) yields a position continuous curve. First (C^1)- and second (C^2)-order continuities imply slope and curvature continuous curves respectively.

A C^1 curve is the minimum acceptable curve for engineering design. Fig. 2.11 shows a geometrical interpretation of these orders of continuity.

A cubic polynomial is the minimum order polynomial that can guarantee the generation of C^0, C^1, or C^2 curves. In addition, the cubic polynomial is the lowest-degree polynomial that permits inflection within a curve segment and that allows representation of nonplanar (twisted) three dimensional curves in space.

Higher-order polynomials are not commonly used in CAD/CAM because they tend to oscillate about control points, are computationally inconvenient and are uneconomical of storing curve and surface representations in the computer.

The type of input data and its influence on the control of the resulting synthetic curve determine the use and effectiveness of the curve in design.

For example, curve segments that require positions of control points and/or tangent vectors at these points are easier to deal with and gather data for than those that might require curvature information.

Also, the designer may prefer to control the shape of the curve locally instead of globally by changing the control point(s). If changing a control point results in changing the curve locally in the vicinity of that point, local control of the curve is achieved; otherwise global control results.

spline, Bezier and B-spline curves. The cubic spline curve passes through the data points and therefore is an interpolant.

Bezier and B-spline curves in general approximate the data points, that is, they do not pass through them. Under certain conditions, the B-spline curve can be an interpolant. Both the cubic spline and Bezier curves have a first-order continuity and the B-spline curve has a second-order continuity.

Hermite Cubic Spline Curve:

They are used to interpolate the given data but not to design free-form curves. Splines derive their name from "French curves or splines"

It connects two data (end) points and utilizes a cubic equation.

Four conditions are required to determine the coefficients of the equation – two end points and the two tangent vectors at these two points.

It passes through the control points and therefore it is an Interpolant. It has only up to C_1 continuity.

The curve cannot be modified locally, i.e., when a data point is moved, the entire curve is affected, resulting in a global control.

The order of the curve is always constant (cubic), regardless of the number of data points. Increase in the number of data points increases shape flexibility, however, this requires more data points, creating more splines that are joined together (only two data points and slopes are utilized for each spline).

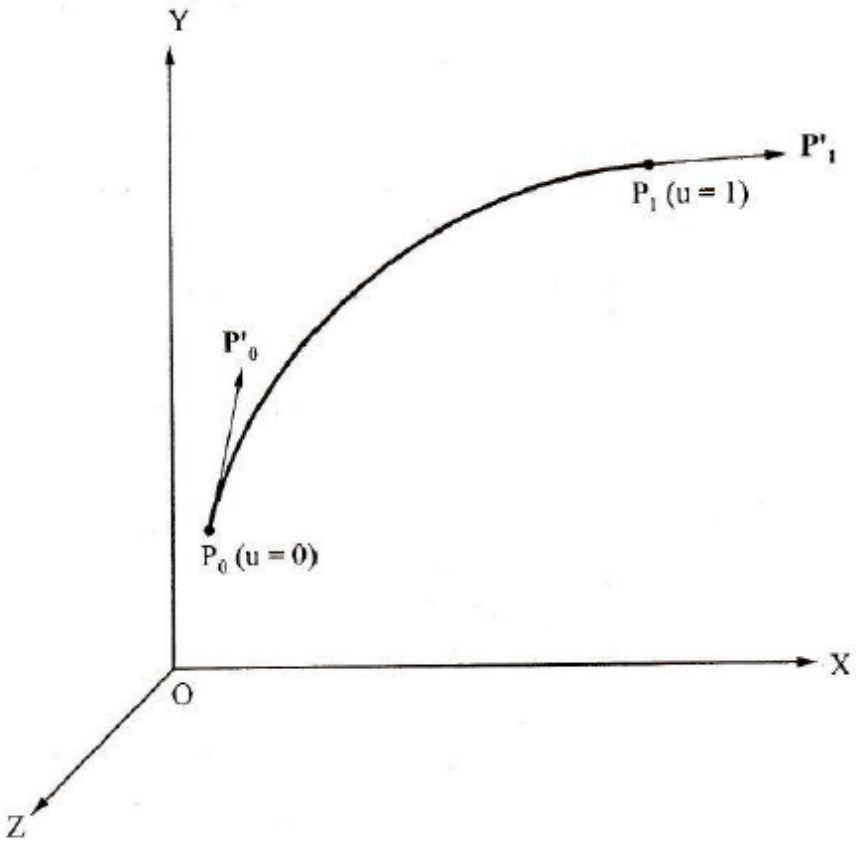


Fig. 2.12 Hermite Cubic Spline Curve

A cubic spline curve utilizes a cubic equation of the following form:

$$P(u)=C_3u^3+C_2u^2+C_1u+C_0 \qquad (2.10)$$

where, $0 \le u \le 1$

The cubic spline is defined by the two endpoints P_0, P_1 and the tangent vectors at these points. The tangent vector can be found by differentiating equ.(2.10).

$$P'(u)=3C_3u^2+2C_2u+C_1 \qquad (2.11)$$

To determine the coefficients of the curve, consider the two endpoints P_0, P_1 and the tangent vectors as shown in Fig. 2.12.

Applying conditions at u = 0, in equations (2.10) and (2.11)

$$P_0=C_0 \qquad (2.12)$$

$$P'_0=C_1 \qquad (2.13)$$

Applying conditions at u = 1, in equations (2.10) and (2.11)

$$P_1=C_3+C_2+C_1+C_0 \qquad (2.14)$$

$$P'_1=3C_3+2C_2+C_1 \qquad (2.15)$$

Substituting (2.12) and (2.13) in (2.14) and (2.15), we get:

$$P_1=C_3+C_2+P'_0+P_0 \qquad (2.16)$$

$$P'_1=3C_3+2C_2+P'_0 \qquad (2.17)$$

By solving simultaneous equations (2.16) and (2.17), we get

$$3P_1-P'_1=C_2+2P'_0+3P_0$$

$$\therefore C_2=3P_1-P'_1-2P'_0-3P_0 \qquad (2.18)$$

Similarly again by solving simultaneous equations (2.16) and (2.17), we get

$$P'_1-2P_1=C_3-P'_0-2P_0$$

$$\therefore C_3=P'_1-2P_1+P'_0+2P_0 \qquad (2.19)$$

Substituting (2.12), (2.13), (2.18) and (2.19) in (2.10), we get:

$$P(u)=(P'_1-2P_1+P'_0+2P_0)u^3+(3P_1-P'_1-2P'_0-3P_0)u^2+P'_0u+P_0$$

$$\therefore P(u)=P'_1u^3-2P_1u^3+P'_0u^3+2P_0u^3+3P_1u^2-P'_1u^2-2P'_0u^2-3P_0u^2+P'_0u+P_0$$

$$\therefore P(u)=2P_0u^3-3P_0u^2+P_0-2P_1u^3+3P_1u^2+P'_0u^3-2P'_0u^2+P'_0u+P'_1u^3-P'_1u^2$$

$$\therefore P(u)=P_0(2u^3-3u^2+1)+P_1(-2u^3+3u^2)+P'_0(u^3-2u^2+u)+P'_1(u^3-u^2) \qquad (2.20)$$

The expression can be written in matrix form as under:

$$P(u)=\begin{bmatrix}P_0 & P_1 & P'_0 & P'_1\end{bmatrix}\begin{bmatrix}2u^3-3u^2+1\\-2u^3+3u^2\\u^3-2u^2+u\\u^3-u^2\end{bmatrix}$$

$$\therefore P(u)=\begin{bmatrix}P_0 & P_1 & P'_0 & P'_1\end{bmatrix}\begin{bmatrix}2&-3&0&1\\-2&3&0&0\\1&-2&1&0\\1&-1&0&0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

On similar lines, the tangent vector equation can be determined by differentiating equation (2.20)

$$P'(u)=P_0(6u^2-6u)+P_1(-6u^2+6u)+P'_0(3u^2-4u+1)+P'_1(3u^2-2u)$$

$$\therefore P'(u)=\begin{bmatrix}P_0 & P_1 & P'_0 & P'_1\end{bmatrix}\begin{bmatrix}6u^2-6u\\-6u^2+6u\\3u^2-4u+1\\3u^2-2u\end{bmatrix}$$

$$\therefore P'(u)=\begin{bmatrix}P_0 & P_1 & P'_0 & P'_1\end{bmatrix}\begin{bmatrix}0&6&-6&0\\0&-6&6&0\\0&3&-4&1\\0&3&-2&0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

Changing the values of endpoints or tangent vectors would modify the shape of the curve.

Example 2.5: A cubic spline curve has start point $P_0(16,0)$ and end point $P_1(3,1)$. The tangent vector for end point P_0 is given by line joining P_0 and point $P_2(14,8)$. Tangent vector for end P_1 is given by line joining P_2 and point P_1.

1. Determine the parametric equation of hermite cubic curve.
2. Plot the hermite cubic curve.

$P_0 = [16 \;\; 0]$

$P_1 = [3 \;\; 1]$

$P'_0 = P_2 - P_0 = [14 \;\; 8] - [16 \;\; 0]$

$= [-2 \;\; 8]$

$P'_1 = P_1 - P_2 = [3 \;\; 1] - [14 \;\; 8]$

$= [-11 \;\; -7]$

Parametric equation

For any point on curve

$$P(u) = \begin{bmatrix} P_0 & P_1 & P'_0 & P'_1 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

X coordinate of a point on curve

$$P_x(u) = \begin{bmatrix} P_{0x} & P_{1x} & P'_{0x} & P'_{1x} \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

$$P_x(u) = \begin{bmatrix} 16 & 3 & -2 & -11 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

$$P_x(u)=\begin{bmatrix}13 & -24 & -2 & 16\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_x(u)=13u^3-24u^2-2u+16$$

Y coordinate of a point on curve

$$P_y(u)=\begin{bmatrix}P_{0y} & P_{1y} & P'_{0y} & P'_{1y}\end{bmatrix}\begin{bmatrix}2 & -3 & 0 & 1\\-2 & 3 & 0 & 0\\1 & -2 & 1 & 0\\1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=\begin{bmatrix}0 & 1 & 8 & -7\end{bmatrix}\begin{bmatrix}2 & -3 & 0 & 1\\-2 & 3 & 0 & 0\\1 & -2 & 1 & 0\\1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=\begin{bmatrix}-1 & -6 & 8 & 0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=-u^3-u^2+8u$$

Thus, parametric equation of parametric curve is

$$P_x(u)=13u^3-24u^2-2u+16$$
$$P_y(u)=-u^3-u^2+8u \qquad (2.21)$$

Example 2.6: The end point of a cubic spline curve are $P_0(1,2)$ and $P_1(7,1)$. The tangent vector for end P_0 is given by line joining P_0 and point $P_2(-2,1)$. The tangent vector for end P_0 is given by line joining $P_3(9,-2)$ and point P_1.

1. Determine the parametric equation of hermite cubic curve.
2. Determine the parametric equation for tangent vector.
3. Plot the hermite cubic curve.

$P_0 = [1 \;\; 2]$

$P_1 = [7 \;\; 1]$

$P'_0 = P_2 - P_0 = [-2 \;\; 1] - [1 \;\; 2] = [-3 \;\; -1]$

$P'_1 = P_1 - P_3 = [7 \;\; 1] - [9 \;\; -2] = [-2 \;\; 3]$

Parametric equation of curve

For any point on curve

$$P(u) = \begin{bmatrix} P_0 & P_1 & P'_0 & P'_1 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

X coordinate of a point on curve

$$P_x(u) = \begin{bmatrix} P_{0x} & P_{1x} & P'_{0x} & P'_{1x} \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

$$P_x(u) = \begin{bmatrix} 1 & 7 & -3 & -2 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

$$P_x(u)=\begin{bmatrix}-17 & 26 & -3 & 1\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_x(u)=-17u^3+26u^2-3u+1$$

Y coordinate of a point on curve

$$P_y(u)=\begin{bmatrix}P_{0y} & P_{1y} & P'_{0y} & P'_{1y}\end{bmatrix}\begin{bmatrix}2 & -3 & 0 & 1\\-2 & 3 & 0 & 0\\1 & -2 & 1 & 0\\1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=\begin{bmatrix}2 & 1 & -1 & 3\end{bmatrix}\begin{bmatrix}2 & -3 & 0 & 1\\-2 & 3 & 0 & 0\\1 & -2 & 1 & 0\\1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=\begin{bmatrix}4 & -4 & -1 & 2\end{bmatrix}\begin{bmatrix}u^3\\u^2\\u\\1\end{bmatrix}$$

$$P_y(u)=4u^3-4u^2-u+2$$

Thus, parametric equation of parametric curve is

$$P_x(u)=-17u^3+26u^2-3u+1$$
$$P_y(u)=4u^3-4u^2-u+2 \qquad (2.22)$$

Parametric equation of tangent vector

$$P'(u)=\begin{bmatrix}P_0 & P_1 & P'_0 & P'_1\end{bmatrix}\begin{bmatrix}0 & 6 & -6 & 0\\ 0 & -6 & 6 & 0\\ 0 & 3 & -4 & 1\\ 0 & 3 & -2 & 0\end{bmatrix}\begin{bmatrix}u^3\\ u^2\\ u\\ 1\end{bmatrix}$$

X coordinate of tangent vector

$$P'_x(u)=\begin{bmatrix}1 & 7 & -3 & -2\end{bmatrix}\begin{bmatrix}0 & 6 & -6 & 0\\ 0 & -6 & 6 & 0\\ 0 & 3 & -4 & 1\\ 0 & 3 & -2 & 0\end{bmatrix}\begin{bmatrix}u^3\\ u^2\\ u\\ 1\end{bmatrix}$$

$$P'_x(u)=\begin{bmatrix}0 & 51 & 52 & -3\end{bmatrix}\begin{bmatrix}u^3\\ u^2\\ u\\ 1\end{bmatrix}$$

Y coordinate of tangent vector

$$P'_y(u)=\begin{bmatrix}2 & 1 & -1 & 3\end{bmatrix}\begin{bmatrix}0 & 6 & -6 & 0\\ 0 & -6 & 6 & 0\\ 0 & 3 & -4 & 1\\ 0 & 3 & -2 & 0\end{bmatrix}\begin{bmatrix}u^3\\ u^2\\ u\\ 1\end{bmatrix}$$

$$P'_y(u)=\begin{bmatrix}0 & 12 & -8 & -1\end{bmatrix}\begin{bmatrix}u^3\\ u^2\\ u\\ 1\end{bmatrix}$$

$$P'_y(u)=12u^2-8u-1$$

Thus, parametric equation of tangent vector is

$$P'_x(u)=51u^2+52u-3$$

$$P'_y(u)=12u^2-8u-1$$

Bezier Curves:

Cubic splines are based on interpolation techniques. Curves resulting from these techniques pass through the given points.

Another alternative to create curves is to use approximation techniques which produce curves that do not pass through the given data points. Instead, these points are used to control the shape of the resulting curves.

Most often, approximation techniques are preferred over interpolation techniques in curve design due to the added flexibility and the additional intuitive feel provided by the former. Bezier and B-spline curves are examples based on approximation techniques.

As its mathematics show shortly, the major differences between the Bezier curve and the cubic spline curve are:

The shape of Bezier curve is controlled by its defining points only. First derivatives are not used in the curve development as in the case of the cubic spline. This allows the designer a much better feel for the relationship between input (points) and output (curve).

The order or the degree of Bezier curve is variable and is related to the number of points defining it; n + 1 points define an nth degree curve which permits higher-order continuity. This is not the case for cubic splines where the degree is always cubic for a spline segment.

The Bezier curve is smoother than the cubic spline because it has higher order derivatives.

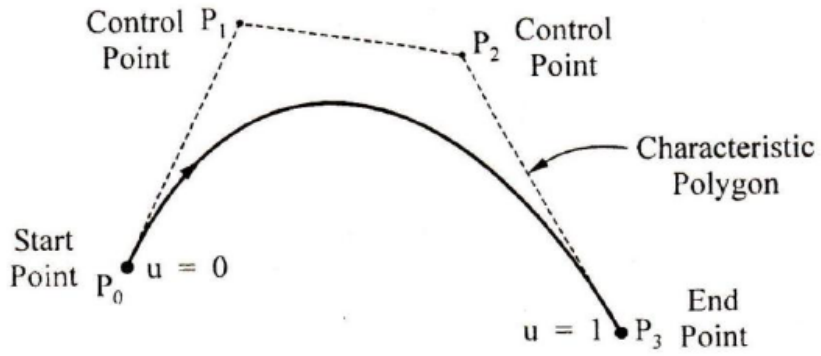


Fig. 2.15 Beizer Curve

The Bezier curve is defined in terms of (n+1) points. This points are called as control points, where n is the degree of the curve.

If n=3 control points are 4, as shown in the Fig 2.15.

These control points form the vertices of the control polygon or Bezier characteristic polygon.

The control or Bezier characteristic polygon uniquely defines the curve.

Properties of Beizer Curve:

The degree of polynomial defining the curve segment is one less than the no. of defining polygon points (n-1).

The curve follows the shape of the defining polygon.

Only the first and last control points or vertices of polygon actually lie on curve.

The other vertices define order & shape of the curve. (see Fig. 3.8)

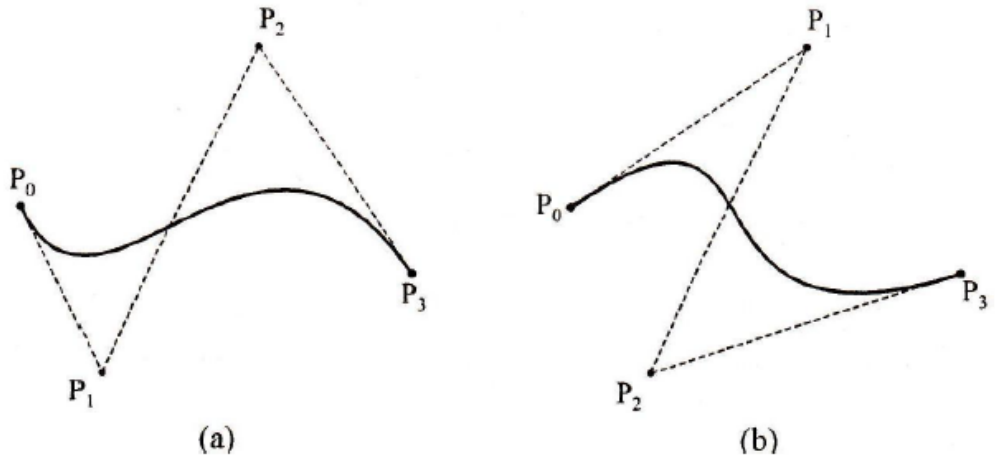


Fig. 2.16 Same data points but different orders for Beizer Curve

The curve is also always tangent to first and last segments of polygon.

The same Beizer curve would be generated, if the sequence of the points is changed from $P_0 - P_1 - P_2 - P_3$ to $P_3 - P_2 - P_1 - P_0$.

The flexibility of the shape would increase with increase in number of vertices of the polygon.

It is having global control on the shape of the curve.

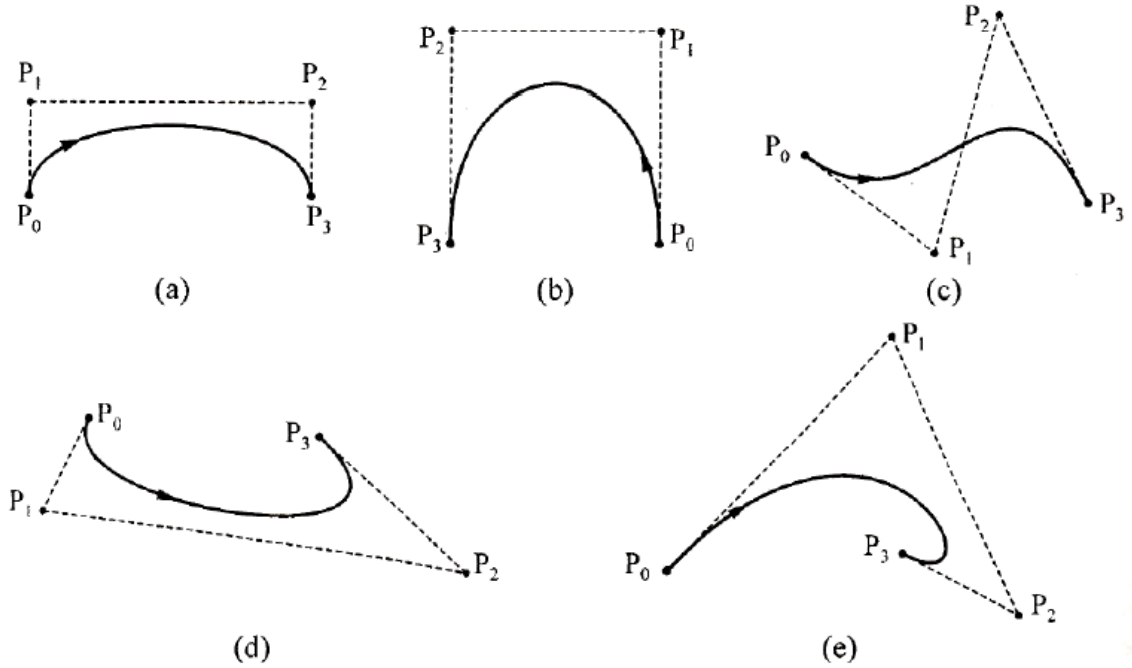


Fig. 2.17 Same Typical Examples of Beizer Curve

Mathematically, for n + 1 control points, the Bezier curve is defined by the following polynomial of degree n:

$$P(u)=\sum_{i=0}^{n} P_i B_{i,n}(u), \quad 0 \le u \le 1 \tag{2.23}$$

where P(u) is any point on the curve and P_i is a control point. $B_{i,n}$ are the Bernstein polynomials. Thus, the Bezier curve has a Bernstein basis. The Bernstein polynomial serves as the blending or basis function for the Bezier curve and is given by

$$B_{i,n}(u)=C(n,i)u^i(1-u)^{n-i} \tag{2.24}$$

where C(n,i) is the binomial coefficient

$$C(n,i)=\frac{n!}{i!(n-i)!} \tag{2.25}$$

Utilizing Eqs. (2.24) and (2.25) and observing that C(n, 0) = C(n, n) = 1,
Eq. (2.23) can be expanded to give

$$P(u)=P_0(1-u)^n+P_1C(n,1)u(1-u)^{n-1}+P_2C(n,2)u^2(1-u)^{n-2}$$
$$+.....+P_{n-1}C(n,n-1)u^{n-1}(1-u)+P_nu^n, \quad 0 \le u \le 1$$

Bezier curve with 3 control points

$$P(u)=P_0(1-u)^2+P_1C(2,1)u(1-u)^{2-1}+P_2C(2,2)u^2(1-u)^{2-2}$$
$$P(u)=(1-u)^2P_0+2u(1-u)P_1+u^2P_2$$

Bezier curve with 4 vertices

$$P(u)=P_0(1-u)^3+P_1C(3,1)u(1-u)^{3-1}+P_2C(3,2)u^2(1-u)^{3-2}+P_3C(3,3)u^3(1-u)^{3-3}$$
$$P(u)=(1-u)^3P_0+3u(1-u)^2P_1+3u^2(1-u)P_2+u^3P_3$$

Bezier curve with 5 vertices

$$P(u)=P_0(1-u)^4+P_1C(4,1)u(1-u)^{4-1}+P_2C(4,2)u^2(1-u)^{4-2}+P_3C(4,3)u^3(1-u)^{4-3}+P_4C(4,4)u^4(1-u)^{4-4}$$
$$P(u)=(1-u)^4P_0+4u(1-u)^3P_1+6u^2(1-u)^2P_2+4u^3(1-u)P_3+u^4P_4$$

Example 2.7: A Bezier curve is to be constructed using control points $P_0(35,30)$, $P_1(25,0)$, $P_2(15,25)$ and $P_3(5,10)$. The Bezier curve is anchored at P_0 and P_3. Find the equation of the Bezier curve and plot the curve for u = 0, 0.2, 0.4, 0.6, 0.8 and 1.

$P_0 = [35 \quad 30]$

$P_1 = [25 \quad 0]$

$P_2 = [15 \quad 25]$

$P_3 = [5 \quad 10]$

$n = 3$

The parametric equation for Bezier curve

$$P(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3$$

X-coordinate of a point on curve

$$P_x(u) = (1-u)^3 P_{0x} + 3u(1-u)^2 P_{1x} + 3u^2(1-u)P_{2x} + u^3 P_{3x}$$

$$P_x(u) = 35(1-u)^3 + 75u(1-u)^2 + 45u^2(1-u) + 5u^3$$

$$P_x(u) = 35(1-3u+3u^2-u^3) + 75u(1-2u+u^2) + 45u^2 - 45u^3 + 5u^3$$

$$P_x(u) = 35 - 105u + 105u^2 - 35u^3 + 75u - 150u^2 + 75u^3 + 45u^2 - 45u^3 + 5u^3$$

$$P_x(u) = 35 - 30u$$

Y-coordinate of a point on curve

$$P_y(u) = (1-u)^3 P_{0y} + 3u(1-u)^2 P_{1y} + 3u^2(1-u)P_{2y} + u^3 P_{3y}$$

$$P_y(u) = 30(1-u)^3 + 75u^2(1-u) + 10u^3$$

$$P_y(u) = 30(1-3u+3u^2-u^3) + 75u^2 - 75u^3 + 10u^3$$

$$P_y(u) = 30 - 90u + 165u^2 - 95u^3$$

The parametric equation of Bezier curve

$$P_x(u) = 35 - 30u$$

$$P_y(u) = 30 - 90u + 165u^2 - 95u^3 \qquad (2.26)$$

The points on curve obtained by varying the value of u = 0, 0.2, 0.4, 0.6, 0.8 and 1 in equ. (2.26) are shown in below table

Point No.	0	2	4	6	8	10
u	0	0.2	0.4	0.6	0.8	1
P_x (u)	35.0	29.0	23.0	17.0	11.0	5.00
P_y (u)	30.0	17.84	14.32	14.88	14.96	10.00

Example 2.8: Plot a Bezier curve using the following control points.

(2,0), (4,3), (5,2), (4,-2), (5,-3) and (6,-2).

$P_0 = [2\ 0]$ $P_1 = [4\ \ 3]$

$P_2 = [5\ 2]$ $P_3 = [4\ \ -2]$

$P_4 = [5\ -3]$ $P_5 = [6\ \ -2]$

n = 5

The parametric equation for Bezier curve with 6 control points

$$P(u) = (1-u)^5 P_0 + 5u(1-u)^4 P_1 + 10u^2(1-u)^3 P_2 + 10u^3(1-u)^2 P_3 + 5u^4(1-u)P_4 + u^5 P_5$$

$$P(u) = (1-5u+10u^2-10u^3+5u^4-u^5)P_0 + 5u(1-4u+6u^2-4u^3+u^4)P_1$$
$$+10u^2(1-3u+3u^2-u^3)P_2 + 10u^3(1-2u+u^2)P_3 + 5u^4(1-u)P_4 + u^5 P_5$$

$$P(u) = (1-5u+10u^2-10u^3+5u^4-u^5)P_0 + (5u-20u^2+30u^3-20u^4+5u^5)P_1$$
$$+(10u^2-30u^3+30u^4-10u^5)P_2 + (10u^3-20u^4+10u^5)P_3 + (5u^4-5u^5)P_4 + u^5 P_5$$

$$P(u) = P_0 + (-5P_0+5P_1)u + (10P_0-20P_1+10P_2)u^2 + (-10P_0+30P_1-30P_2+10P_3)u^3$$
$$+(5P_0-20P_1+30P_2-20P_3+5P_4)u^4 + (-P_0+5P_1-10P_2+10P_3-5P_4+P_5)u^5$$

X-coordinate of a point on a curve

$$P_x(u) = P_{0x} + (-5P_{0x}+5P_{1x})u + (10P_{0x}-20P_{1x}+10P_{2x})u^2 + (-10P_{0x}+30P_{1x}-30P_{2x}+10P_{3x})u^3$$
$$+(5P_{0x}-20P_{1x}+30P_{2x}-20P_{3x}+5P_{4x})u^4 + (-P_{0x}+5P_{1x}-10P_{2x}+10P_{3x}-5P_{4x}+P_{5x})u^5$$

$$P_x(u) = 2 + (-5(2)+5(4))u + (10(2)-20(4)+10(5))u^2 + (-10(2)+30(4)-30(5)+10(4))u^3$$
$$+(5(2)-20(4)+30(5)-20(4)+5(5))u^4 + (-(2)+5(4)-10(5)+10(4)-5(5)+(6))u^5$$

$$P_x(u) = 2+10u-10u^2-10u^3+25u^4-11u^5$$

Y-coordinate of a point on a curve

$$P_y(u) = P_{0y} + (-5P_{0y}+5P_{1y})u + (10P_{0y}-20P_{1y}+10P_{2y})u^2 + (-10P_{0y}+30P_{1y}-30P_{2y}+10P_{3y})u^3$$
$$+(5P_{0y}-20P_{1y}+30P_{2y}-20P_{3y}+5P_{4y})u^4 + (-P_{0y}+5P_{1y}-10P_{2y}+10P_{3y}-5P_{4y}+P_{5y})u^5$$

$$P_y(u) = 0 + (-5(0)+5(3))u + (10(0)-20(3)+10(2))u^2 + (-10(0)+30(3)-30(2)+10(-2))u^3$$
$$+(5(0)-20(3)+30(2)-20(-2)+5(-3))u^4 + (-(0)+5(3)-10(2)+10(2)-5(-3)+(-2))u^5$$

$$P_y(u) = 15u-40u^2+10u^3+25u^4-12u^5$$

The parametric equation of Bezier curve

$$P_x(u) = 2+10u-10u^2-10u^3+25u^4-11u^5$$
$$P_y(u) = 15u-40u^2+10u^3+25u^4-12u^5 \qquad (2.27)$$

The points on curve obtained by varying the value of u from 0 to 1 in steps of 0.1 in equ.(2.27) are shown in below table

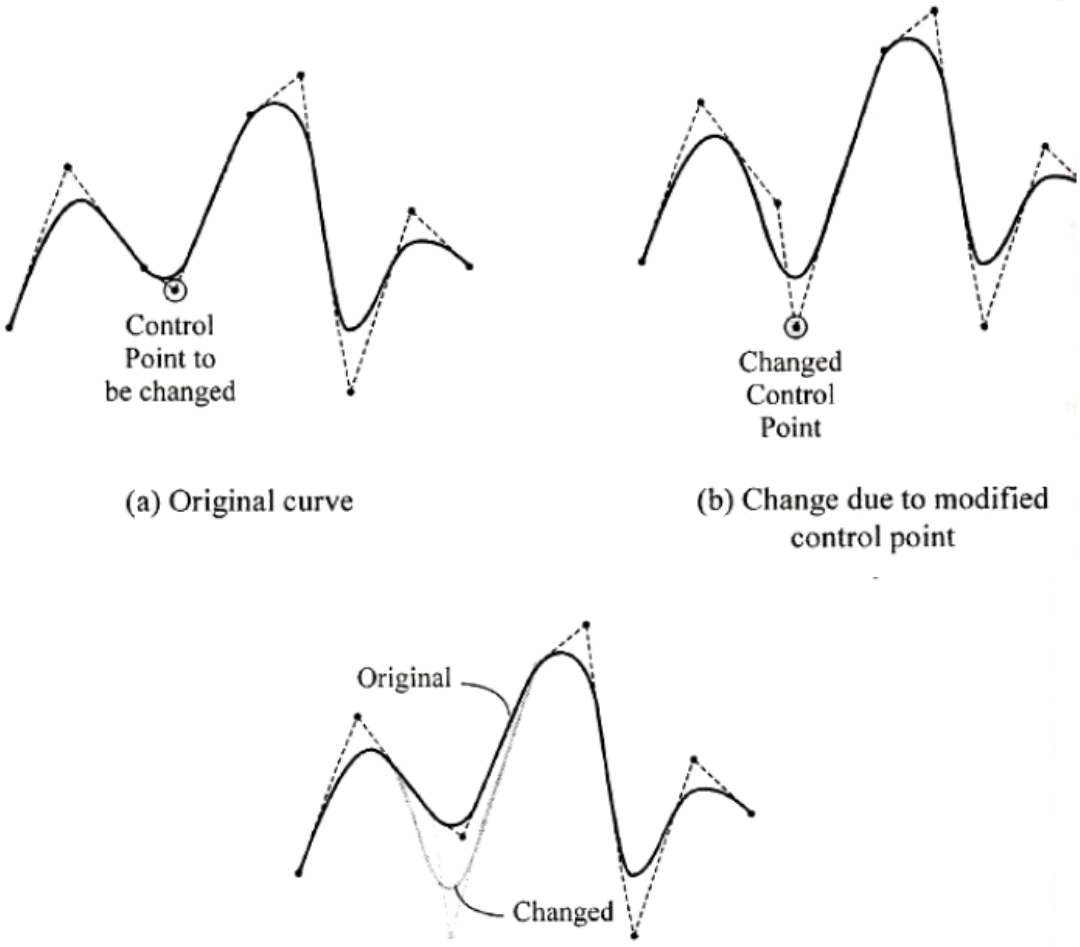
Point No.	0	1	2	3	4	5	6	7	8	9	10
u	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
P_x (u)	2	2.89	3.56	4.01	4.29	4.47	4.62	4.82	5.12	5.52	6.00
P_y (u)	0	1.11	1.52	1.34	0.76	-0.06	-0.93	-1.68	-2.17	-2.29	-2.00

B-Spline Curve:

B-Spline curves are proper and powerful generalization of Bezier curve.

They are very similar to Bezier curve and are also defined with the help of characteristic polygon.

The major advantage of B-Spline curve is due to its ability to control the curve shape locally, as opposed to global control (see Fig. 2.20).



(a) Original curve

(b) Change due to modified control point

(c) Both curves superimposed

Fig. 2.20 Local Control of a B-Spline Curve

Rational curves

A rational curve is defined by the algebraic ratio of two polynomials while a nonrational curve is defined by one polynomial.

The formulation of rational curves requires the introduction of homogeneous space and the homogeneous coordinates. The homogeneous space is four-dimensional space. A point in three dimensional with coordinates (x, y, z) is represented in the homogeneous space by the coordinates (x*, y*, z*, h),where h is a scalar factor. The relationship between the two types of coordinates is

$$x = \frac{x^*}{h} \qquad y = \frac{y^*}{h} \qquad z = \frac{z^*}{h}$$

A rational B-spline curve defined by n + 1 control points P_i is given by

$$P(u) = \sum_{i=0}^{n} P_i R_{i,k}(u), \quad 0 \le u \le u_{max} \tag{2.29}$$

$R_{i,k}(u)$ are the rational B-spline functions and are given by

$$R_{i,k}(u) = \frac{h_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)}$$

The above equation shows that $R_{i,k}(u)$ are a generalization of the nonrational basis functions $N_{i,k}(u)$. If we substitute $h_i = 1$ in the equation, $R_{i,k}(u) = N_{i,k}(u)$. The rational basis functions $R_{i,k}(u)$ have nearly all the analytic and geometric characteristics of their nonrational B-spline counterparts.

The main difference between rational and nonrational B-spline curves is the ability to use h_i at each control point to control the behavior of the rational B-splines (or rational curves in general).

UNIT-III

Surface Modeling

The surface model is an extension of wire frame model which involves providing a surface representation making the object look solid to the viewer.

Surface systems were in fact the first CAD systems to raise the possibility of integrating the whole industrial process of design and analysis through to the next stages of production and quality control, using the computer as an intermediary.

Shape design and representation of complex objects such as car, ship, aircraft and castings cannot be achieved by a wire frame model. In such cases, surface models are preferred for representing the objects with precision and accuracy.

These models also find widespread applications in companies manufacturing forgings, castings and molded products. Such articles are characterized by smoothly curved shapes with blended edges which are not easily represented by conventional engineering drawings.

Another difficulty for designers with wire frame modelers is to obtain good visualization of their ideas. The surface modeler is ideal for such products as the entire surface is represented; allowing the computer to generate very realistic shaded surface pictures.

Most surface modelers come equipped with *rendering* features. In this, the model once created is then provided with surface properties. For example, the surface may be given a property which may make the object appear corroded or made of brass or any such effects.

A huge material library is generally made available to select the desired surface effect. One can also define different kinds of lights, such as spot lights, ambient lights, etc. in a 3D space and generate a photorealistic image of the object. This feature is highly used in automobile and styling industries for determining the aesthetic appeal of the object being designed.

Advanced surface modelers go beyond representation and to some extent can be used for property calculations as well. The surface model can be used for generating NC tool paths for continuous path machining, making it an ideal mode for integrating CAD / CAM. A surface modeler can be considered as an extension of a wire frame.

A wire frame model can be easily extracted from a surface model. It should be remembered that surface models only store the geometry of their corresponding objects and not the topology of these objects. To generate a surface model, a user typically starts with a wire frame model and then connects them with the desired surfaces.

Once the product has been surface modeled, its geometry may be used directly in the design of the mould or die which is to be made. Essential calculations can usually be performed automatically by the computer such as for example the determination of the volume enclosed by a mould. This gives an immediate idea of the volume of raw material needed to make the product.

Kinds of Surfaces

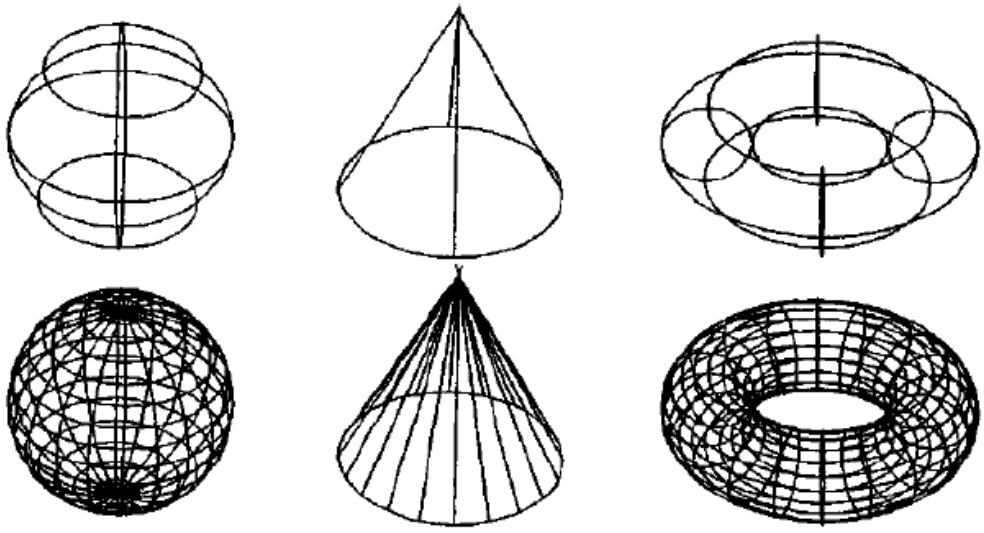


Fig. 2.21 Effect of Mesh Size on Object Visualization

To create a surface, data related to the desired shape is required. The choice of the surface is dependent on the application. In order to visualize surfaces, a *mesh* of *m* x *n* in size is displayed. The mesh size is controllable and is in the form of criss-cross on the surface. The mesh size plays an important role in proper visualization as can be seen in the objects shown in *Fig. 2.21*. However, it should be noted that finer the mesh size of surface entities, longer is the time required by the software to construct and modify the entities.

A system may need two boundaries to create a ruled surface or might require one entity to create a surface of revolution. Surfaces provided by CAD / CAM software are either *analytical* or *synthetic.*

The following are descriptions of major surface entities provided by CAD/CAM systems:

1. **Plane Surface:** This is the simplest surface. It requires three noncoincident points to define an infinite plane. The plane surface can be used to generate cross-sectional views by intersecting a surface model with it, generate cross sections for mass property calculations, or other similar applications where a plane is needed. Fig. 2.22 shows a plane surface.

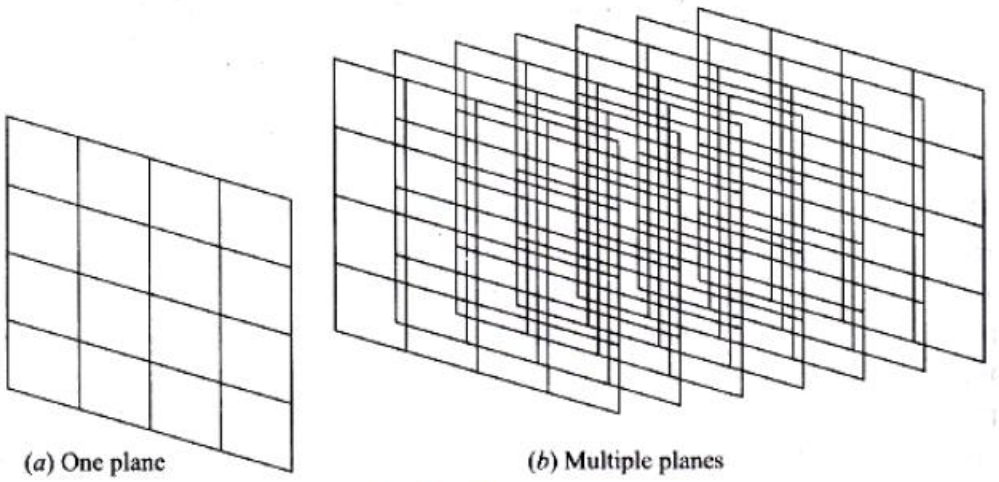


Fig. 2.22 Plane Surface

2. **Ruled (lofted) Surface:** This is a linear surface. It interpolates linearly between two boundary curves that define the surface (rails). Rails can be any wireframe entity. This entity is ideal to represent surfaces that do not have any twists or kinks. Fig. 2.23 gives some examples.

3. **Surface of Revolution:** This is an axisymmetric surface that can model axisymmetric objects. It is generated by rotating a planar wireframe entity in space about the axis of symmetry a certain angle (Fig. 2.24).

4. **Tabulated Cylinder:** This is a surface generated by translating a planar curve a certain distance along a specified direction (axis of the cylinder) as shown in Fig. 2.25. The plane of the curve is perpendicular to the axis of the cylinder. It is used to generate surfaces that have identical curved cross sections. The word "tabulated" is borrowed from the APT language terminology.

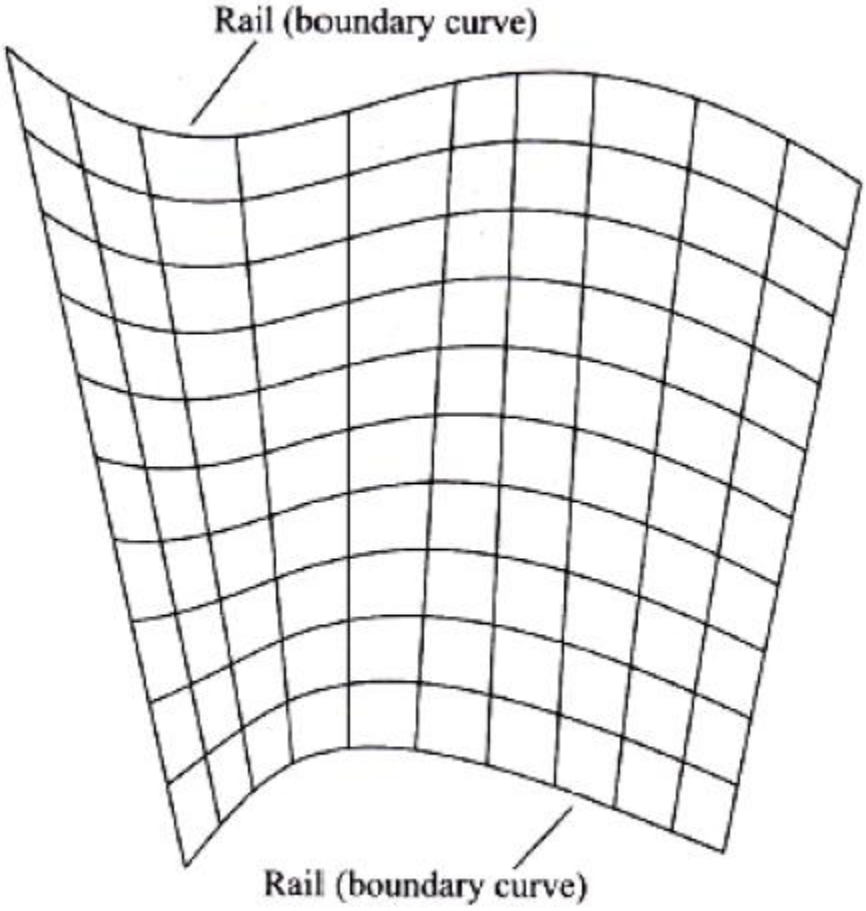


Fig. 2.23 Ruled Surface

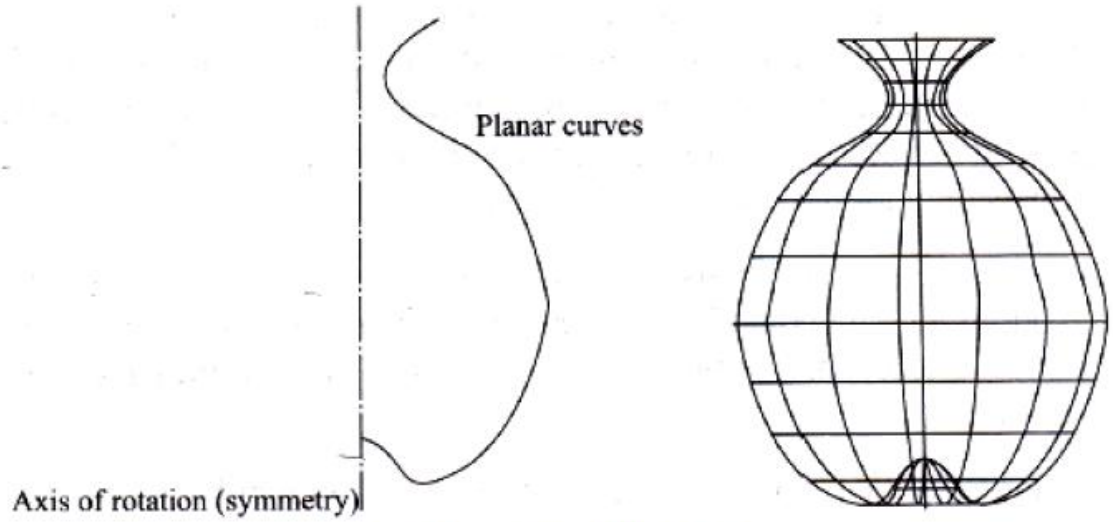


Fig. 2.24 Surface of Revolution

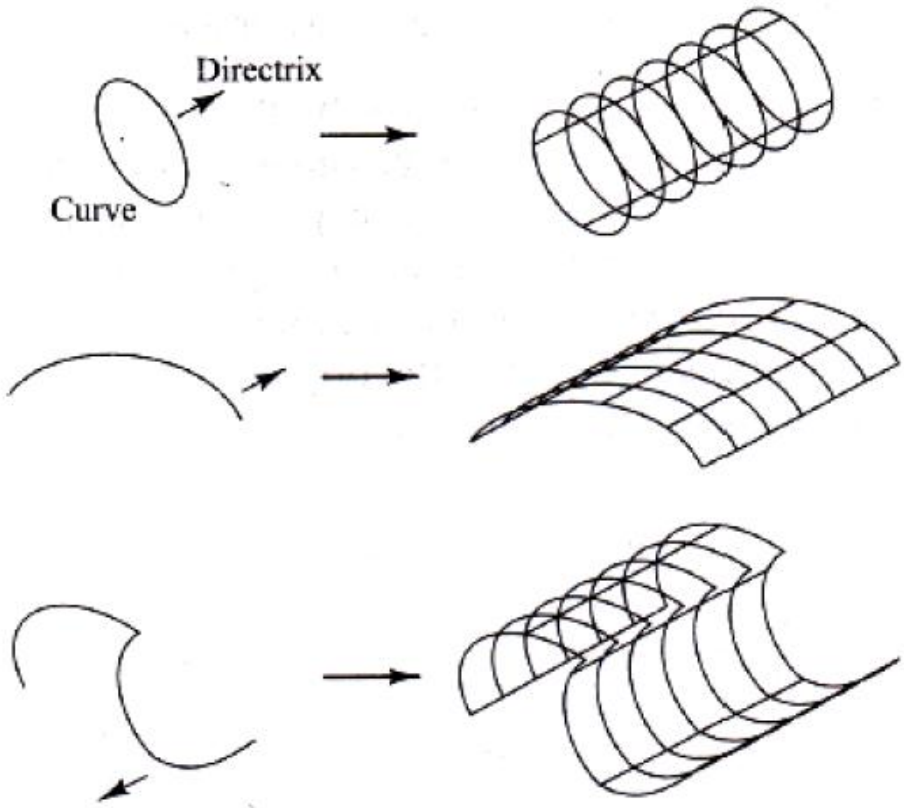


Fig. 2.25 Tabulated Cylinder

5. **Bezier surface:** This is a surface that approximates given input data. It is different from the previous surfaces in that it is a synthetic surface. Similarly to the Bezier curve, it does not pass through all given data points. It is a general surface that permits twists and kinks (Fig. 2.26). The Bezier surface allows only global control of the surface.

6. **B-spline surface:** This is a surface that can approximate or interpolate given input data (Fig. 2.27). It is a synthetic surface. It is a general surface like the Bezier surface but with the advantage of permitting local control of the surface.

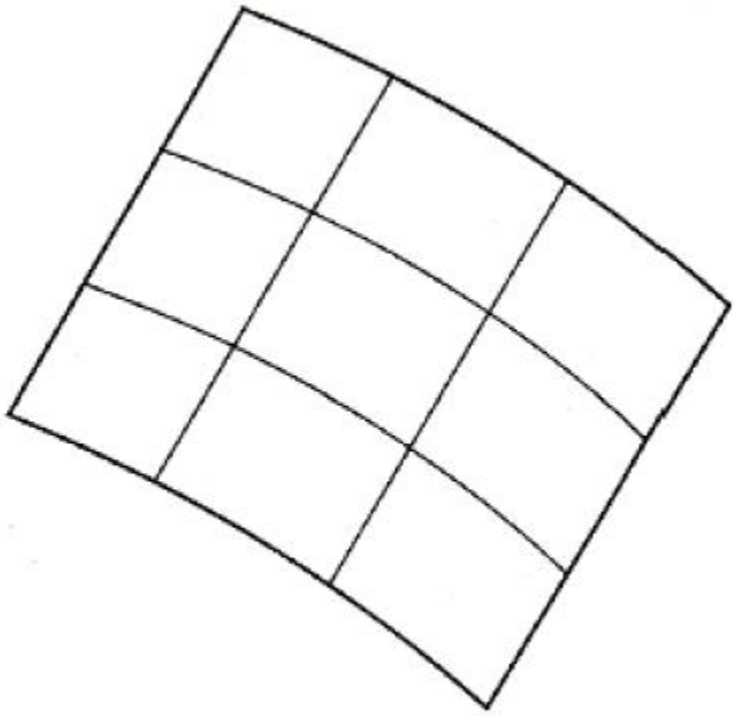
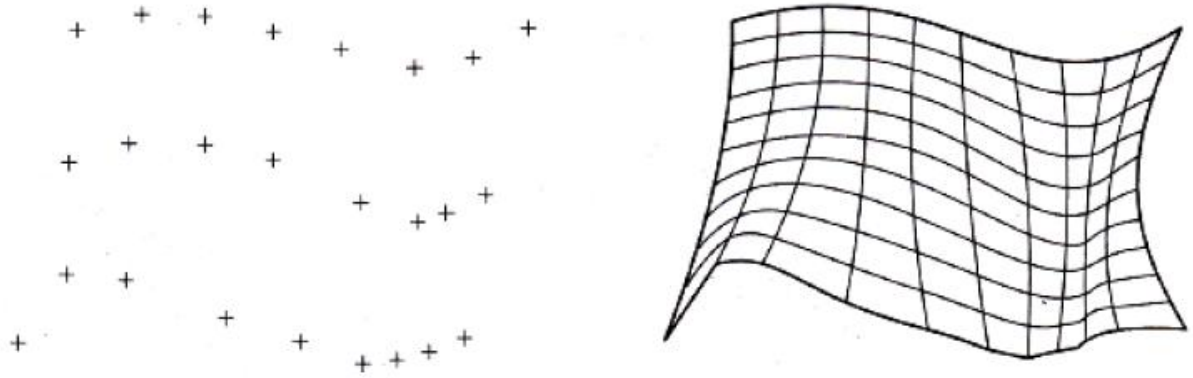


Fig. 2.26 Bezier Surface



(*a*) Data points (*b*) B-spline surface

Fig. 2.27 B-spline Surface

7. **Coons Patch:** The above surfaces are used with either open boundaries or given data points. The Coons patch is used to create a surface using curves that form closed boundaries (Fig. 2.28).

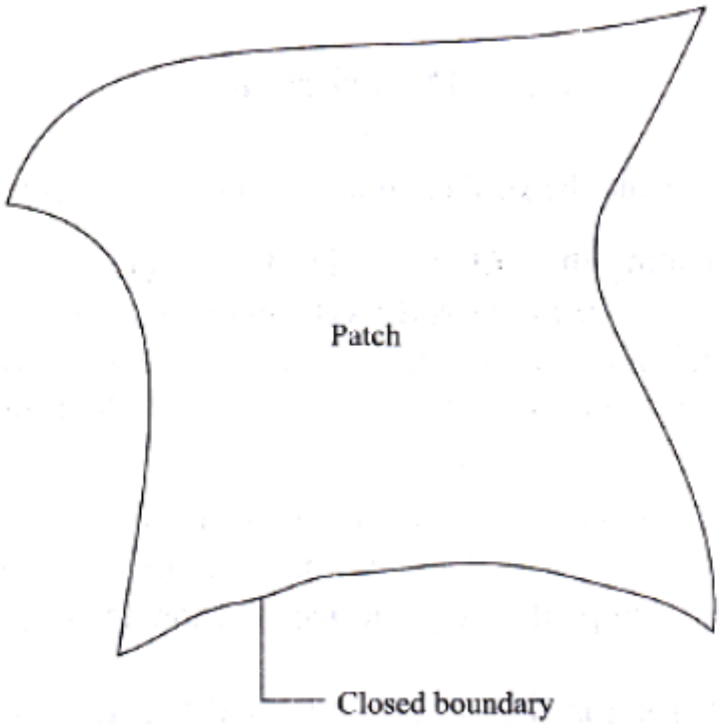


Fig. 2.28 Coons Patch

8. **Fillet surface:** This is a B-spline surface that blends two surfaces together (Fig. 2.29). The two original surfaces may or may not be trimmed.

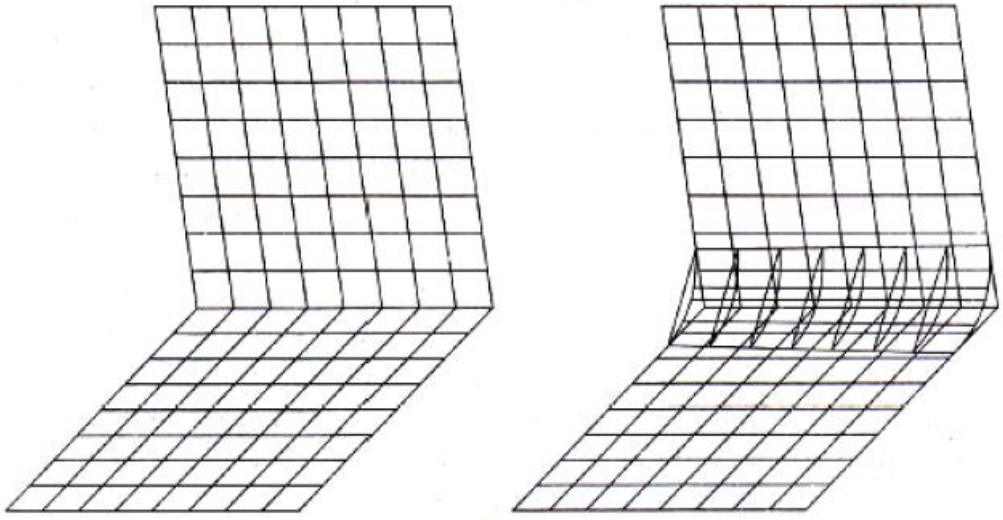


Fig. 2.29 Fillet Surface

9. **Offset surface:** Existing surfaces can be offset to create new ones identical in shape but may have different dimensions. It is a useful surface to use to speed up surface construction. For example, to create a hollow cylinder, the outer or inner cylinder can be created using a cylinder command and the other one can be created by an offset command. Offset surface command becomes very efficient to use if the original surface is a composite one. Figure 2.30 shows an offset surface.

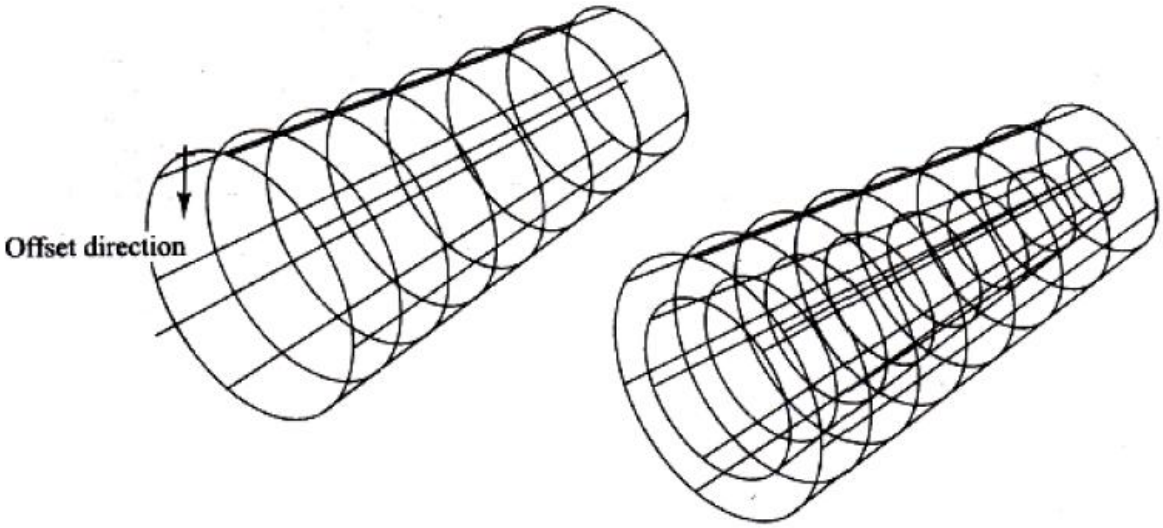


Fig. 2.30 Offset Surface

Unit IV

Surfaces

- Many objects we want to model are not flat:
  - Cars, animals, plants, buildings.

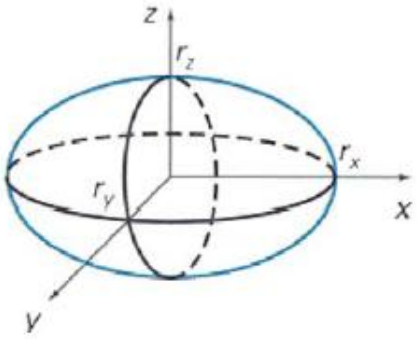
Surface Representations

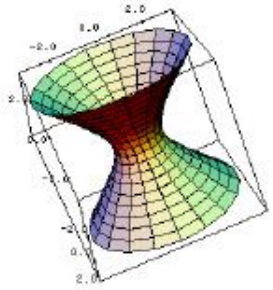
- Parametric: (x,y,z)=(f(u,v), g(u,v), h(u,v))
  - e.g. plane, sphere, cylinder, torus, bi-cubic surface, swept surface
  - parametric functions let you *iterate* over the surface by incrementing u and v
  - great for making polygon meshes, etc
  - complex for intersections: ray/surface, point-inside-boundary, etc
- Implicit: F(x,y,z) = 0
  - e.g. plane, sphere, cylinder, quadric, torus, blobby models
  - terrible for iterating over the surface
  - great for intersections, morphing

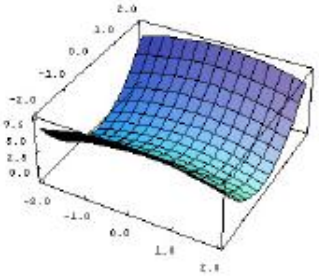
Examples

Paremetric: Ellipsoid

$$x = r_x \cos\phi \cos\theta$$
$$y = r_y \cos\phi \sin\theta$$
$$z = r_z \sin\phi$$



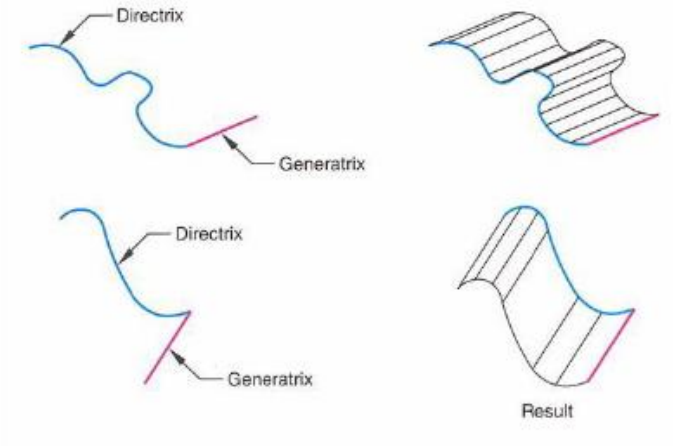




Implicit functions:
Quadrics and other

Swept Surfaces

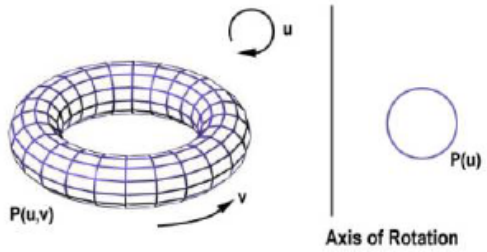
Obtained by sweeping generator enities along director entities.



Rotational Surfaces

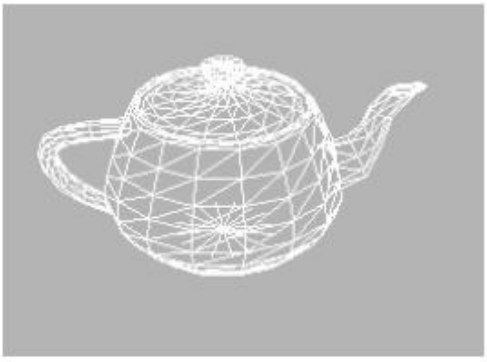
Generated by rotating a curve about an axis.

Every point of the generating curve describes a circle whose supporting plane lies orthogonally to the Axis.



Meshes

We can approximate a surface with a *polygonal mesh.*

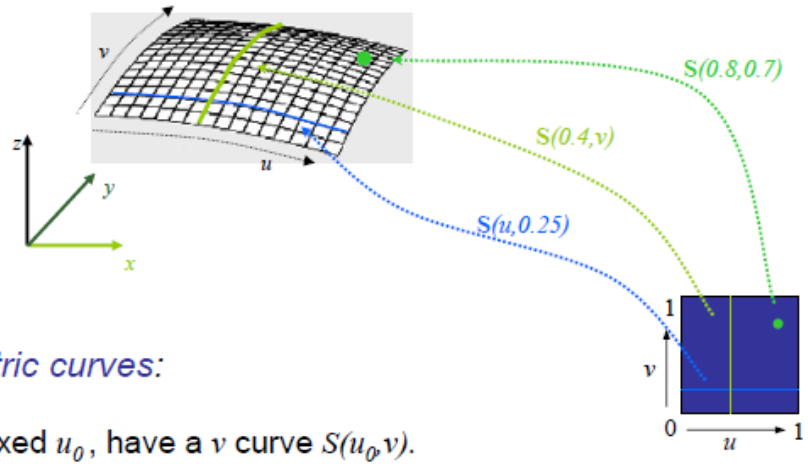


Curved Surfaces

- Remember the overview of curves:
  - Described by a series of control points.
  - A function $Q(t)$.
  - Segments joined together to form a longer curve.
- Same for surfaces, but now two dimensions
  - Described by a mesh of control points.
  - A function $S(u,v)$.
  - *Patches* joined together to form a bigger surface.

Parametric Surface Patch

- $S(u,v)$ describes a point in space for any given (u,v) pair:
  - u,v each range from 0 to 1.



- *Parametric curves:*
  - For fixed u_0, have a v curve $S(u_0,v)$.
  - For fixed v_0, have a u curve $S(u,t_0)$.
  - For any point on the surface, there are a pair of parametric curves that go through point.

Polynomial Surface Patches

- $S(s,t)$ is typically polynomial in both s and t
  - *Bilinear:*

$$S(s,t) = \mathbf{a}st + \mathbf{b}s + \mathbf{c}t + \mathbf{d}$$

$$S(s,t) = (\mathbf{a}t + \mathbf{b})s + (\mathbf{c}t + \mathbf{d}) \quad \text{-- hold } t \text{ constant} \Rightarrow \text{linear in } s$$

$$S(s,t) = (\mathbf{a}s + \mathbf{c})t + (\mathbf{b}s + \mathbf{d}) \quad \text{-- hold } s \text{ constant} \Rightarrow \text{linear in } t$$

  - *Bicubic:*

$$S(s,t) = \mathbf{a}s^3t^3 + \mathbf{b}s^3t^2 + \mathbf{c}s^3t^2 + \mathbf{d}s^3 + \mathbf{e}s^2t^3 + \mathbf{f}s^2t^2 + \mathbf{g}s^2t + \mathbf{h}s^2 + \mathbf{i}st^3 + \mathbf{j}st^2 + \mathbf{k}st + \mathbf{l}s + \mathbf{m}t^3 + \mathbf{n}t^2 + \mathbf{o}t + \mathbf{p}$$

$$S(s,t) = (\mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d})s^3 + (\mathbf{e}t^3 + \mathbf{f}t^2 + \mathbf{g}t + \mathbf{h})s^2 + (\mathbf{i}t^3 + \mathbf{j}t^2 + \mathbf{k}t + \mathbf{l})s + (\mathbf{m}t^3 + \mathbf{n}t^2 + \mathbf{o}t + \mathbf{p}) \quad \text{-- hold } t \text{ constant} \Rightarrow \text{cubic in } s$$

$$S(s,t) = (\mathbf{a}s^3 + \mathbf{e}s^2 + \mathbf{i}s + \mathbf{m})t^3 + (\mathbf{b}s^3 + \mathbf{f}s^2 + \mathbf{j}s + \mathbf{n})t^2 + (\mathbf{c}s^3 + \mathbf{g}s^2 + \mathbf{k}s + \mathbf{o})t + (\mathbf{d}s^3 + \mathbf{h}s^2 + \mathbf{l}s + \mathbf{p}) \quad \text{-- hold } s \text{ constant} \Rightarrow \text{cubic in } t$$

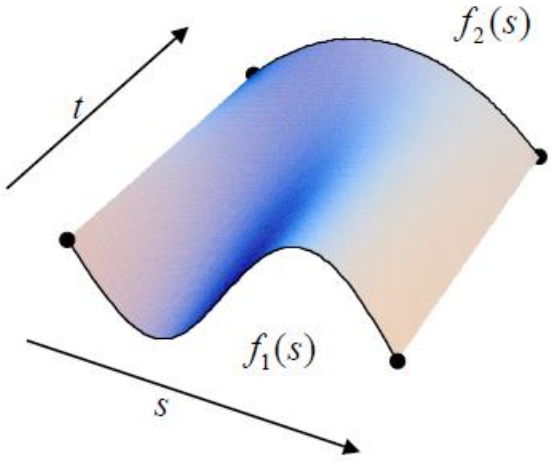
Properties of the bilinear patch

- Interpolates the control points.
- The boundaries are straight line segments connecting the control points.
- If the all 4 points of the control mesh are co-planar, the patch is flat.
- If the points are not coplanar, get a curved surface.
- The parametric curves are all straight line segments:
  - Is a (doubly) *ruled surface:* has (two) straight lines through every point.

Ruled Surfaces

- Linear interpolation between 2 curves
  - All point lie in one line

$$f(s,t) = (1-t) f_1(s) + t\, f_2(s)$$



Particular case

Cylinder:

$$f(s,t) = f_1(s) + t\, d$$

$$f_1(s) = (\cos s, \sin s, 0)$$

Cone:

$$f(s,t) = (1-t) f_1 + t\, f_2(s)$$

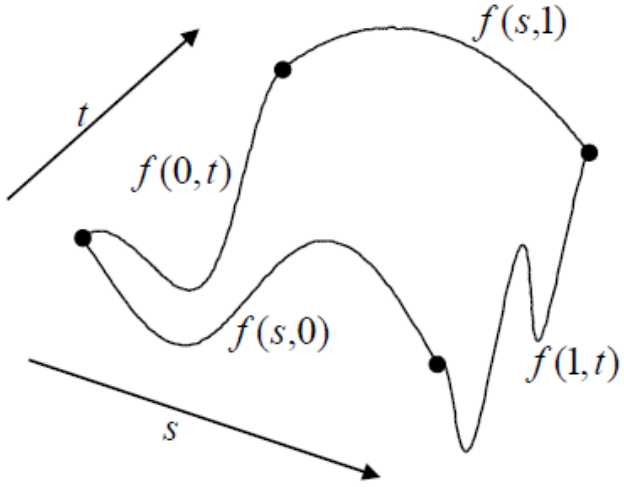
$$f_2(s) = (\cos s, \sin s, 0)$$

Coons patches

- Interpolation between 4 curves
- Build a ruled surface between pairs of curves

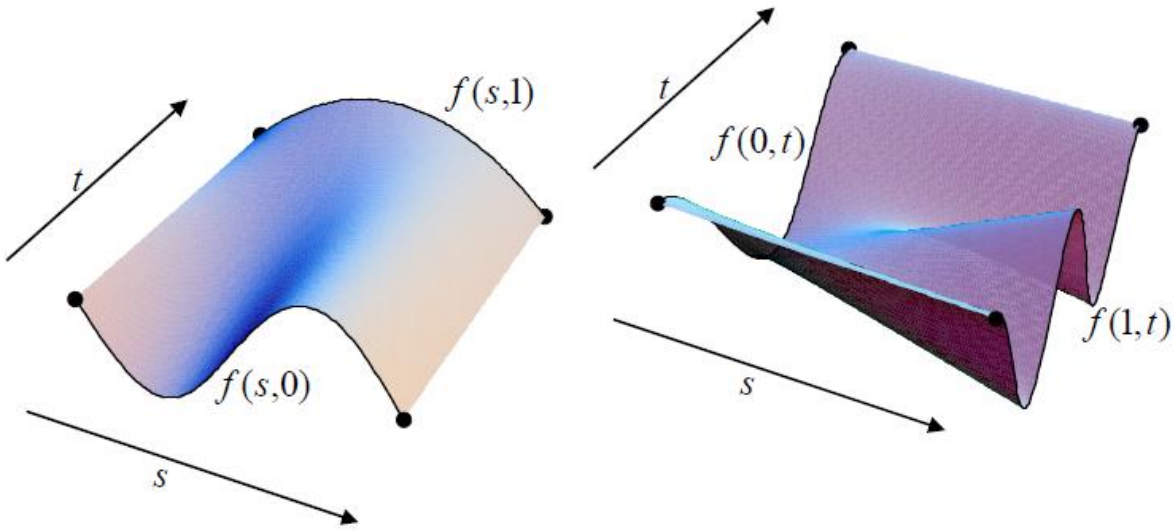
$$f_1(s,t) = (1-t) f(s,0) + t\, f(s,1)$$

$$f_2(s,t) = (1-s) f(0,t) + s\, f(1,t)$$



Coons patches **Step 2**

$$f_1(s,t) = (1-t)\,f(s,0) + t\,f(s,1) \qquad f_2(s,t) = (1-s)\,f(0,t) + s\,f(1,t)$$



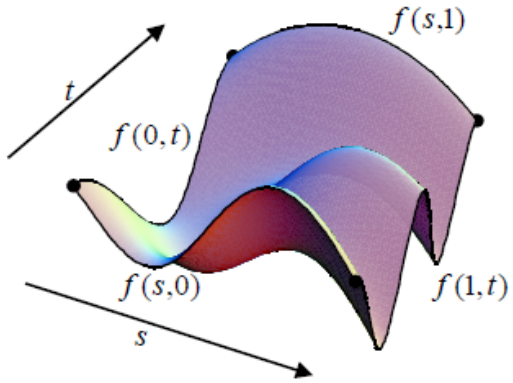
Coons patches **Step 3**

- "Correct" surface to make boundaries match
  - Create a linear interpolation surface between the 4 extremes:

$$f_3(s,t) = (1-s)t\,f(0,1) + s(1-t)\,f(1,0) + s\,t\,f(1,1))$$ (bilinear patch).

  - Combine surface as $f_1+f_2-f_3$

$$f_1(s,t) + f_2(s,t) - ((1-s)(1-t)\,f(0,0) + (1-s)t\,f(0,1) + s(1-t)\,f(1,0) + s\,t\,f(1,1))$$

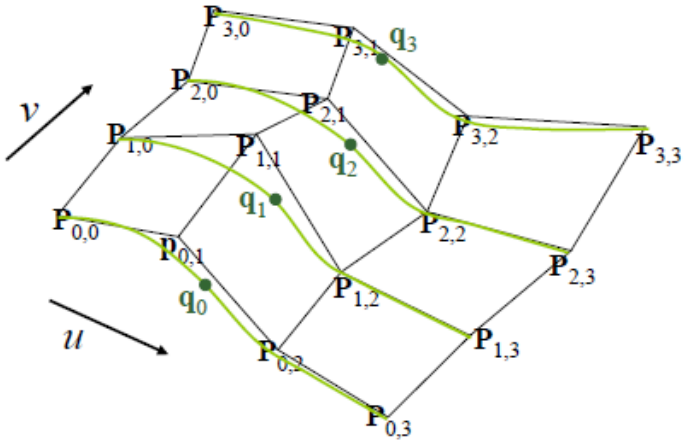


Bezier Control Mesh

- A bicubic patch has a grid of 4x4 control points:

$$P_{0,0}, \ldots, P_{0,3}$$
$$\ldots$$
$$P_{3,0}, \ldots, P_{3,3}.$$

- Defines four Bézier curves along u and four Bézier curves along v.
- Evaluate using same approach as bilinear.



Bezier Patch Properties

- Convex hull: any point on the surface will fall within the convex hull of the control points.
- Interpolates 4 corner points.
- Approximates other 12 points, which act as "handles".
- The boundaries of the patch are the Bézier curves defined by the points on the mesh edges.
- The parametric curves are all Bézier curves.

Cubic Bezier Surfaces: Algebraic Formulation

Cubic Bezier curves can be extended to surfaces on unit squares:

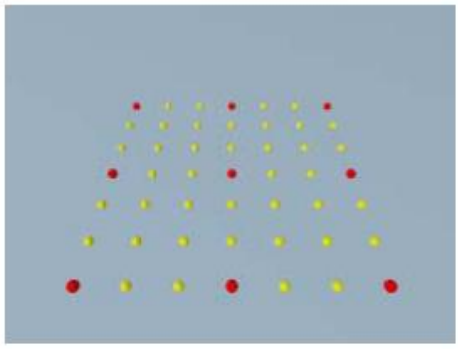
$$S(u,v) = \sum_{i=0}^{i=3} \sum_{j=o}^{j=3} B_i^3(u) B_j^3(v) P_{ij}$$

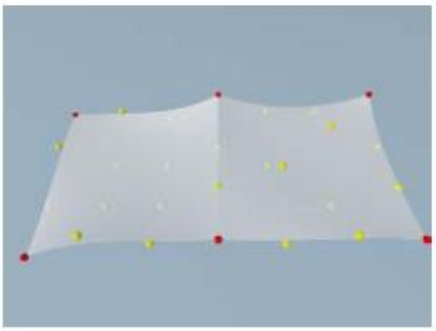
$$B_i^3(t) = \binom{3}{i} t^i (1-t)^{3-i}$$

Building a surface from Bezier patches

Building complex surfaces by putting together Bezier patches.

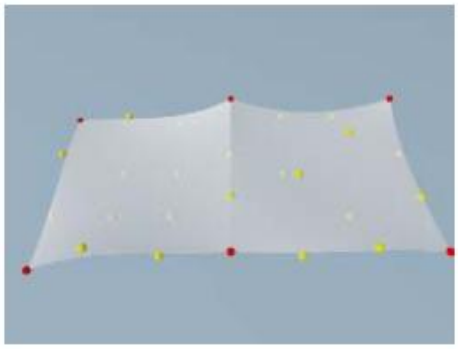
- Lay out grid of adjacent meshes.
- For C^0 continuity, must share points on the edge
  - Each edge of a Bézier patch is a Bézier curve based only on the edge mesh points.
  - So if adjacent meshes share edge points, the patches will line up exactly.

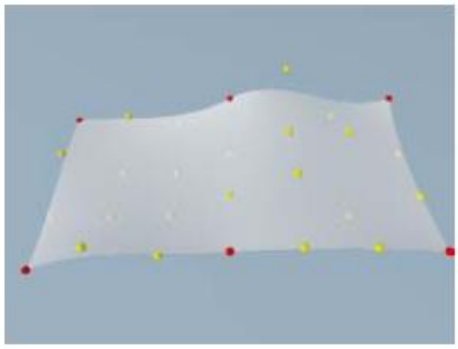




C^1 continuity across Bézier edges

- We want the parametric curves that cross each edge to have C^1 continuity:
  - So the handles must be equal-and-opposite across the edge.





B-spline patches

For the same reason as using B-spline curves:

- More uniform behavior.
- Better mathematical properties.
- Doesn't interpolate any control points.

B-Spline Surfaces

A B-spline surface $S(u,v)$, is defined by:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n'} B_{i,d}(u) B_{j,d'}(v) P_{i,j}$$

where:

- The $P_{i,j}$ are the $(n+1)$ x $(n'+1)$ control points.
- d and d' are the orders in the u and v directions.
- We have two non-decreasing *knot sequences of parameters* $u_0,\ldots,u_{n+d}$ and $v_0,\ldots,v_{n'+d'}$.
- $B_{i,d}$ are the uniform B-Spline *basis* or *blending functions* of degree $d-1$.

NURBS Surfaces

$$S(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n'} w_{i,j} B_{i,d}(u) B_{j,d'}(v) P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{n'} w_{i,j} B_{i,d}(u) B_{j,d'}(v)}$$

- Can take on more shapes:
  - conic sections.
- Can blend, merge, … .
- Still has rectangular topology.

Blending of surfaces

Basic idea of blending surfaces

A space curve or a space curve segment (resp. surface) is called *regular* if a nonzero tangent vector (resp. normal vector) at every point on the curve (resp. surface) exists and is unique.

Let S_1, S_2 be two surfaces, $C = S_1 \bigcap S_2$ a regular space curve. If for each point $P \in C$, the tangent plane of S_1 at P is the same as the tangent plane of S_2 at P, then S_1 and S_2 are said to meet with G^1-*continuity* along C.

Now consider the following problem.

Problem 2.1 Let S_1 and S_2 be two surfaces in real 3-Eudidean space R^3, $C_1 \subset S_1, C_2 \subset S_2$ regular space curves called boundary curves. To construct a surface S which connects S_1, S_2 along C_1, C_2 respectively with G^1-continuity.

Theorem 2.2 Let S_1, S_2 be two surfaces, $C = S_1 \bigcap S_2$ a regular space curve. If for each $P \in C$, there exists a regular space curve $C_1 \subset S_1$, such that $P \in C_1$ and

$$N \cdot T_1 = 0, T_1 \neq a * T, a \in \mathbf{R},$$

then S_1 meets S_2 with G^1-continuity along C. Here T denotes the tangent vector of C at P, N denotes the normal vector of S_2 at P, and T_1 denotes the tangent vector of C_1 at P.

Proof. Since $T_1 \neq a * T$, a normal vector of S_1 at point P is $T \times T_1$. Since $N \cdot T_1 = 0$, T_1 is perpendicular to N. Since C is in S_2 and N is the normal vector of S_2 at point P, N is perpendicular to T. This means N and $T \times T_1$ are proportional. As a result, the tangent planes of S_1 and S_2 at point P is the same. So the theorem holds.

Theorem 2.2 tells us how to construct blending surfaces when the normal vectors on the boundary curves are known or can be gotten, as in the case of implicit surfaces.

Theorem 2.3 Let S_1, S_2 be two surfaces, $C = S_1 \bigcap S_2$ a regular space curve. For each $P \in C$, suppose that there exist regular space curves $C_1 \subset S_1$,$C_2 \subset S_2$ such that $P \in C_1$, $P \in C_2$. Furthermore, suppose that C_1 connects C_2 at point P with G^1-continuity, and the tangent vector of C_2 (or C_1) at point P is not parallel to the tangent vector of C at P. Then S_1 and S_2 meet with G^1 continuity along C.

Proof. Let T_1, T_2 be the tangent vectors of C_1 and C_2 at point P, T the tangent vector of C at point P. Since C_1 and C_2 meet at P with G^1-continuity, $T_1 = \alpha T_2, \alpha \in \mathbf{R}$. Note that T_1 (T_2) is not parallel to T. Then the normal vectors of S_1 and S_2 at point P is $N_1 = T_1 \times T = \alpha T_2 \times T$ and $N_2 = T_2 \times T$ respectively. Since $N_1 = \alpha N_2$, the tangent planes of S_1 and S_2 at point P are the same. So the theorem holds.

Theorem 2.3 will be used to construct blending surfaces between parametric surfaces.

Remark 2.4 If the boundary curves are not regular, the blending surface itself will not be G^1-continuity or self-intersects.

Constructing parametric blending surface

In this section, we will show how to blend two surfaces with a parametric surface.

Bézier curves

Let

$$B_{i,n}(t) = \frac{n!}{(n-i)!i!}t^i(1-t)^{n-i}, 0 \le t \le 1$$

denote the *Bernstein polynomials*. A Bézier curve of degree n with control points V_i can be defined as

$$B^n(t) = \sum_{i=0}^{n} V_i B_{i,n}(t)$$

with some properties listed below.

Terminal points:

$$V_0 = B^n(0), V_n = B^n(1).$$

Tangents on terminal points: Let T_0, T_1 be the unit tangent vectors of the Bézier curve at points V_0, V_n respectively. Then the following equations hold:

$$T_0 = \frac{V_1 - V_0}{\|V_1 - V_0\|}, T_1 = \frac{V_n - V_{n-1}}{\|V_n - V_{n-1}\|}.$$

To construct a space curve which passes two points and the tangent vectors of the curve at the two points are appointed, we can use Bézier curve. Choose the two points as terminal control points, and select two points on the given tangent lines as the other two control points. The Bézier curve defined by the four points is what we want.

Method of constructing blending surface

To solve Problem 2.1, we further assume $C_1(s) \subset S_1$, $C_2(s) \subset S_2, s \in [0,1]$ have parametric forms and have been parameterized to have the same variable s. For $s = s_0 \in [0,1]$, we will construct a space curve $f(s_0, t)$ which satisfies the following equations:

$$\begin{cases} f(s_0, 0) = C_1(s_0), \\ f(s_0, 1) = C_2(s_0), \\ \frac{\partial f(s_0,t)}{\partial t}|_{t=0} \cdot N_1(s_0) = 0, \\ \frac{\partial f(s_0,t)}{\partial t}|_{t=1} \cdot N_2(s_0) = 0, \end{cases} \tag{1}$$

where $N_i(s_0)(i = 1, 2)$ denote the normal vectors of S_1, S_2 at $C_1(s_0), C_2(s_0)$ respectively. They may be unknown but only need to ensure that the tangent vector of $f(s_0, t)$ at $t = 0$ (resp. $t = 1$) is in the tangent plane of S_1 (resp. S_2) at point $C_1(s_0)$ (resp. $C_2(s_0)$). When s changes in [0,1], $f(s, t)$ forms a parametric surface. The last two equations are used to guarantee that the tangent vectors of $f(s, t)$ are parallel to tangent planes of S_1 and S_2 at $C_1(s)$ and $C_2(s)$ respectively.

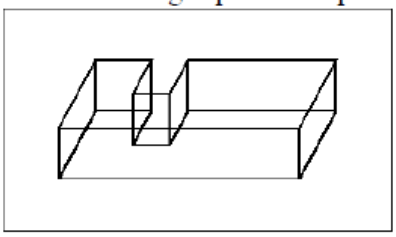
Unit V

CAD/CAM and solid modelling concepts

A key to the understanding of Product Data Exchange is some knowledge of the way geometry is represented in the involved CAD-systems. CAD-systems using a similar internal geometry representation can exchange geometry without much difficulty provided that some neutral file interface exists [3]. On the other hand if the target application of the transfer uses a different representation from the sending system one must try to compromise on the design metodology and export information in a way which can be understood by the target system.

Wireframe representation

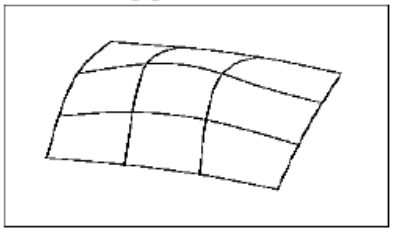
A wireframe model is represented by defining edges and points. An edge may be a line or a curve. This representation is natural for a designer who is familiar with mechanical drawings, since it is the lines and curves in a drawing which define 3D shape. A wireframe model is simple to deal with in a computer with small storage space and quick access time.



Pure wireframe representation is has a number of drawbacks as a basis for modelling 3D solids, notably the possibility of creating ambiguous models and the lack of graphic or visual coherence.

Surface representation

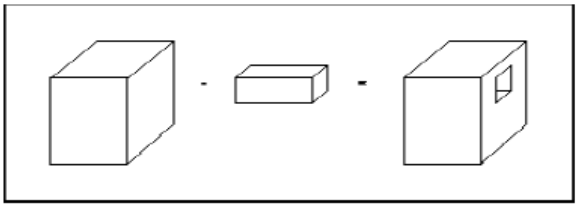
A surface model is represented by edges and points, as is a wireframe model, but with additional *faces* which fills the space between the edges and points. Each face is described by a surface, which can be elements of quadrics like cone, cylinder, sphere, or a set of splines. One of the most common is the B-Spline representation [1]. B-Spline (Basis-Spline) is one category of surfaces employing parametric polynomials using parameters.



In CAD systems for design using free-form surfaces sculptured surfaces, surface models are commonly used for the internal representation. However, a surface model does not contain topological information, and can therefore be ambiguous when determining the volume of an object. Surface models play an important role in industry, because they give an accurate description of the surface of an object which can be used e.g. to guide NC-milling machines or other manufacturing-oriented applications. Another area where free-form surfaces are in extensive use is for styling of e.g. car-bodies and other consumer products where the shape and design plays and important role.

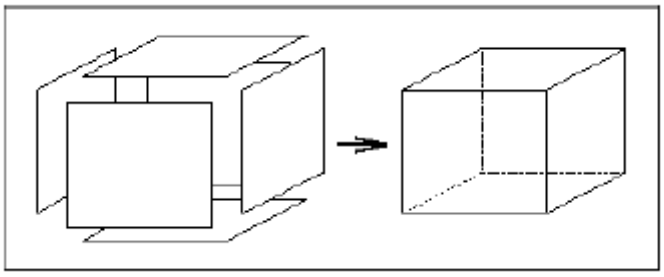
Constructive Solid Geometry (CSG)

With CSG, solids are described as combinations of simple primitives or other solids in a series of Boolean operations, i.e. a CSG model is constructed using a so called building block approach. A user operates on parameterised instances of solid primitives with Boolean operators.



Boundary representation (B-rep)

B-Rep models represent a solid indirectly by a representation of its bounding surface. A B-Rep solid is represented as a volume contained in a set of faces together with topological information which defines the relationships between the faces. Because B-Rep include such topological information, a solid is represented as a closed space in 3D space. The boundary of a solid separates points inside from points outside of the solid. B-rep models can represent a wide class of objects but the data structure is complex, and it requires a large memory space. A very simple B-rep model constructed using 6 faces is shown in Figure .



Boundary representation can be divided in three classes: facetted, elementary, and advanced B-Rep. In facetted B-Rep, a solid is bounded by planar surfaces. Only points, planes and planar polygons are required and are implicitly represented by their vertex points. The surfaces included in elementary B-Rep are planar, quadric, and toroidal surfaces. The bounding curves of the faces are lines, conics, or 4th order curves. In advanced B-Rep, the surfaces includes also spline surfaces (B-Spline, Bézier, NURBS, etc.) in addition to elementary B-Rep. The bounding curves are spline curves.

Hybrid representation

Modern CAD-systems almost always use some hybrid form of representation which is a combination of the basic variations described in the previous text. This is because different representations have their advantages and drawbacks for different application areas. Typically CAD-systems use either CSG- or B-rep and an internal primary representation, while approximated facetted B-rep is used as a secondary representation for display purposes.

Two major approaches exist. In the first approach, the modeller has CSG as the primary representation. From CSG, B-Rep models are created using boundary evaluation. The user has no direct access to the secondary representation. (Examples of this type of modellers are EUCLID[1] and Bravo3[2].) In the second approach, the modeller has B-Rep as the primary representation. Such modellers often use CSG as a way to let the user describe basic shapes and do operations on them via the user interface. However, such systems typically allow additional modification methods which directly modify boundary data structures and thus the CSG representation can not be updated. (Examples are ACIS[3], Parasolid[4] and Pro/ENGINEER[5].)

One should note that conversions from *CSG to B-rep. is possible* by evaluating the CSG tree, while conversion of a *B-rep. model to CSG is not possible* by conventional methods and is still subject for research.

Product data exchange standards & methods

CAD-vendors traditionally tend to protect their share of the marked by sticking to proprietary formats and not put serious efforts into the implementation of neutral interfaces. Typically the vendors try to push their customer to buy their particular analysis module or say, NC-machining preparation software. Market pressure and use of various computer platforms for CAD-packages has broken this trend, although a number of vendors stick to their old habits. Nevertheless quite some exchange can be done via existing formats if a good methodology is used.

Current exchange formats and *de-facto* standards

CAD-exchange formats (Table 1) should be used for exchange of CAD models when there is a need to access the exact geometry. When a model is needed for visualisation and verification purposes, a Graphics format (Table 2) may be more useful and easier to handle. This is especially the case for 2D Drawings, which often are exchanged only to be visually checked. A format like HPGL is then more appropriate than a CAD-exchange format like IGES or DXF.

Table 1: CAD-exchange formats currently in use

Format:	Full name, explanation	Used to exchange:
IGES	Initial Graphics Exchange Specification	Wireframe, Surfaces, and drawings
DXF	Drawing Exchange Format from Autodesk	Drawings and Wireframe
SET	Standart d'Exchange et Transfert	Wireframe, Surfaces, drawings, solids
ACIS SAT	ACIS Solid Exchange Format	ACIS models (Autocad, Microstation etc.)
VDA-FS	VDA, Flachen Schnittstelle	Surface geometry
STEP AP203	ISO 10303-203 Configuration Controlled Design	3D CAD models and product data

Table 2: Graphics exchange formats

Format	Comment
VRML(Virtual Reality Modeling Language)	Standard for viewing 3D objects via the Web rapidly gaining popularity.
Wavefront, Inventor, 3DS, Flight	3D graphics *de-facto* standard formats for Facetted B-rep models
SLA Stereolithography format	Triangular facets used for Rapid Prototyping applications
HPGL, Postscript	Image files used for plotters. Good for visualising drawings.

[1] EUCLID and EUCLID3 are trademarks of Matra Datavision
[2] Bravo and Bravo3 are trademarks of Schlumberger Technologies
[3] ACIS is a trademark of Spatial Technology Inc.
[4] Parasolid is a trademark of Electronic Data Systems Corp. Copyright ©1996 EDS
[5] Pro/ENGINEER is a trademark of Parametric Technology Corporation

STEP

STEP, Standard for the Exchange of Product Model Data, provides a representation of product information along with the necessary mechanisms and definitions to enable product data to be exchanged between different applications and processes.

The overall objective of STEP is to provide a mechanism that is capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product data bases and archiving. The ultimate goal is an integrated product information database that is accessible and useful to all the processes necessary to support a product over its lifecycle.

STEP addresses different computer applications associated with the complete product lifecycle including design, manufacture, utilization, maintenance, and disposal. STEP is thus not only targeting CAD/CAM applications, but includes processes related to the organisation of the product data such as definition of materials, formal contracts and specifications which are valid across organisations.

Conformance to STEP is therefore very important when one implements an Engineering Data Management System, (EDMS or PDM-system) which describes Engineering processes and the organisation of Product Data. STEP provides standard templates and mechanisms for how to describe organisational data related to engineering products without company-specific flavours.

STEP Architecture

STEP is organized as a series of parts, grouped into different series: *description methods*, *integrated resources*, *application protocols*, *abstract test suites*, *implementation methods*, and *conformance testing*.

STEP uses a formal specification language, EXPRESS (Part 11), to specify the product information to be represented. The use of a computerised Data Definition language enables precision and consistency of representation and facilitates development of implementations.

The *Integrated Resources* is a library of general purpose information models for things like geometry, topology, product identification, dates, times etc. (The 40-series parts.)

Among the *Implementation Methods* are a Physical File format (Part 21) and a group of standard application programming interfaces (Part 22-26)

By means of the EXPRESS language and common definitions from the STEP integrated resources, so called *Application Protocols* (APs) are used to specify the representation of product information for one or more industry-specific application area.

Examples of STEP APs are:

- AP203 : Configuration controlled 3D designs of mechanical parts and assemblies
- AP212 : Electrotechnical Plants
- AP214 : Core Data for Automotive Design Processes.
- AP221 : Process Plant Functional Data and Schematic Representation.
- AP 223 :Exchange of Design & Manufacturing Product Information for Cast Parts

IGES

IGES (Initial Graphics Exchange Specification) was the first specification for CAD data exchange published in 1980 as a NBS (National Bureau of Standards) report in USA. IGES version 1.0 was accepted and released in 1981 as an ANSI standard. All major CAD vendors support IGES and it is currently by far the most widespread standard for CAD data exchange.

IGES was originally developed for the exchange of drafting data like 2D/3D wireframe models, text, dimensioning data, and a limited class of surfaces. Due to criticism and bad experience with the data transfer using IGES, the standard has been gradually extended and developed concerning supported entities, syntax, clarity, and consistency. The current version, IGES 5.2, provides the following capabilities:

Geometry: 2D/3D wireframes, 2D/3D curves and surfaces, CSG (since version 4.0 in 1988), B-Rep (since version 5.1 in 1991);
Presentation: Drafting entities for technical drawings;
Application dependent elements: Piping and electronic schematics, AEC elements;
Finite Element Modeling: Elements for FEM systems.

IGES specification defines the format of the file, language format, and the product definition data in these formats. The product definition includes geometric, topological, and non-geometric data. The geometry part defines the geometric entities to be used to define the geometry. The topology part defines the entities to describe the relationships between the geometric entities. The geometric shape of a product is described using these two parts (i.e. geometry and topology). The non-geometric part can be divided into annotation, definition, and organization. The annotation category consists of dimensions, drafting notations, text, etc. The definition category allows users to define specific properties of individual or collections of entities. The organization category defines groupings of geometric, annotation, or property elements.

An IGES file consists of six sections: Flag, Start, Global, Directory Entry, Parameter Data, and Terminate. Each entity instance consists of a directory entry and parameter data entry. The directory entry provides an index and includes attributes to describe the data. The parameter data defines the specific entity. Parameter data are defined by fixed length records, according to the corresponding entity. Each entity instance has bi-directional pointers between the directory entry and the parameter data section.

The size of IGES files and consequently the processing time are practical problems. IGES files are composed of fixed format records and each entity has to have records in both the directory entry section and the parameter data section with bi-directional pointers. This causes also errors in pre- and post-processor implementations.

IGES is under control of the NCGA (National Computer Graphics Association) and is part of the U.S. Product Data Association (USPRO) and the IGES/PDES Organization (IGO). The NCGA administers the National IGES User Group (NIUG), which provides access to information on IGES.

DXF

DXF (Data eXchange Format) was originally developed by Autodesk, Inc., the vendor of AutoCAD. It has become a "de-facto" standard among most CAD vendors and is in wide use to exchange 2D/3D wireframe data. All implementations of AutoCAD accept this format and are able to convert it to and from their internal representation. A DXF file is a complete representation of the AutoCAD drawing database thus some features or concepts can't be used by other CAD systems. The DXF version R13 supports wireframe, surface, and solid representations.

A DXF file consists of four sections: Header, Table, Block, and Entity section. The header section contains general information about the drawing. Each parameter has a variable name and an associated value. The table section contains definitions of line types, layers, text styles, views, etc. The block section contains entities for block definitions. These entities define the blocks used in the drawing. The format of the entities in the block section is identical to entities in the entity section. The entity section contains the drawing entities, including any block references. Items in the entity section exist also in the block section and the appearance of entities in the two sections is identical.

Variables, table entries, and entities are described by a group that introduces the item, giving its type and/or name, followed by multiple groups that supply the values associated with the item. In addition, special groups are used for separators such as markers for the beginning and end of sections, tables, and the file itself. Group codes are used to describe the type of the value, and the general use of the group.

A collaborative design taxonomy

This taxonomy organizes issues into a framework to offer direction in identifying areas that may need investigation. Research into multiple subjects, including engineering design, collaboration, and teamwork, has shown that collaborative design can be described by several attributes. These attributes, which compose the top level of the taxonomy, are: team composition, communication, distribution, design approach, information, and nature of the problem. A description of the effects of the primary collaborative design attributes follows. The sub-levels of the taxonomy for each factor are displayed with each explanation.

1 Team Composition

Extensive research has been conducted in the fields of psychology and sociology to analyze the impact of team composition on effective and timely team performance [10, 11, 12, 13]. Team composition can be divided into characteristics of the group, characteristics of individuals, team member relations, and leadership styles as shown in Table 1.

Table 1. Team composition issues in collaborative design

1. Team Composition	A. Group: *Size, Culture*
	B. Individual: *Personality, Expertise*
	C. Team Member Relations: *Positive, Neutral, Negative*
	D. Leadership Styles: *Autocratic, Consultative, Collective, Participative, Leaderless*

Research suggests that the size of the team should match the complexity of the task [10, 11]. Willaert suggests that teams that are too large may become unmanageable and require additional organizational structure, while creativity may be inhibited if teams are too small. The quantitative effect of a range of group culture variables on innovative productivity was studied by Hurley [12]. He found that group cultures that emphasize participative decision-making, characterized by openness and involvement in decision-making, are associated with higher levels of innovation. The organizational hierarchy, as well as the leadership styles within the design team may influence this area of the group culture.

Diverse theories have been developed relating team members' personalities and the performance of design teams. The five-factor personality model (FFM), which includes conscientiousness, extraversion, stability, agreeableness, and openness to experience, brings some order to the field of personality research [13]. Reilly, et al. [14] proposed relationships between a team's average level of each personality factor and the overall team performance to suggest preferred team composition for particular design types.

The amount of knowledge or experience perceived to be required before making a contribution to the team may affect productivity of younger engineers [15]. Varying levels of experience on a team, however, may facilitate success in innovation by combining the wisdom of age with the energy and idealism of youth [10, 15]. Additionally, the type of design problem may dictate which areas of expertise should be included in the team composition. Including several disciplines in a team seems to both enrich and complicate many areas of the design process [3].

Although engineering design is mainly a technical activity, it truly functions as a social activity. Lloyd proposes that design is a process of building on individual, social, and organizational experiences [16]. If a collaborative design team is distributed, its ability to collectively utilize these experiences may be hindered. Research has found that group cohesiveness is positively related to team success, and team member relations have a major positive relationship with team performance and team satisfaction [17, 18].

In addition to team cohesiveness, leadership styles may also influence collaborative design. The Vroom-Yetton model, which includes autocratic, consultative, collective, participative, and leaderless styles, may be used to classify leadership styles [19]. Austin, et al.'s empirical studies of interdisciplinary teams found that a team needs to be led through the design activity [20]. Further, the team needs to agree on who should lead and what leadership style should be used in order for the group to work effectively. Other research has found positive relationships between leadership style and team performance, work climate, and team learning [18, 21].

2 Communication

By definition, collaborative design teams share expertise, ideas, resources, or responsibilities, which necessitates a strong communication system. The issues outlined in Table 2 impact this sharing process.

Table 2. Communication issues in collaborative design

2. Communication	A. Mode: *Verbal, Written, Graphic, Gestures*
	B. Quantity: *Frequency, Duration*
	C. Syntax: *Common Language, Translators*
	D. Proficiency of Team: *Techniques, Technology*
	E. Dependability of Resources: *Reliability, Availability*
	F. Intent: *Inform, Commit, Guide, Request, Express, Decide, Propose, Respond, Record*

The form of communication that is chosen at various collaborative design interfaces may facilitate or hinder the process. It follows that some communication forms (verbal, written, graphic, or gestures) may be better suited for use in particular tasks or phases of the design process. Team members' selection of particular communication forms may be influenced or even governed by the perceived importance of the information to be communicated, dispersion of the team over time and space, the task in the design process, the effort required to use each method, team composition, and other factors [22, 23]. Collaboration technologies are necessary to overcome the inherent resistance to the flow of information encountered by distributed design teams [24].

The proficiency of the team in using various communication tools might influence form selection, frequency, and success [23]. Effective communication may be inhibited if team members use various languages in their communication. This includes spoken and written languages, as well as information and query languages. The needs of agents receiving information may determine the required syntax or view of the information [25]. If communication systems are not reliable, low user satisfaction will likely negate enhanced functional capabilities the systems are intended to provide [26].

A list of ten communicative actions (inform, commit, guide, request, express, decide, propose, respond, and record) has been developed to represent the intent of communication [27]. Some modes of communication and information forms may have more than one purpose.

3 Distribution

In collaborative design, the teams members and information may be collocated, but are more likely to be distributed across some variety of boundaries, (geographic, organizational, temporal) as shown in Table 3.

Dispersion of team members may have a significant impact on the team's choice of communication techniques, frequency of communication, and language [20]. Research to compare the graphic communication of distributed teams to those of collocated teams showed that remote designers spent 51% more time making graphic acts than their collocated counterparts [28]. However, the production of sketches, which are considered important because they impose order while stimulating reinterpretation, decreased significantly when teams were distributed. Some research has found that collocation had neither direct nor indirect effects on project outcomes [29].

Table 3. Distribution issues in collaborative design

3. Distribution	A. Personnel: *Collocated, Geographically Distributed, Distributed Across Organizations, Distributed Across Time Boundaries*
	B. Information: *Collocated, Geographically Distributed, Distributed Across Organizations, Distributed Across Time Boundaries*

The availability of communication resources may also be inhibited, primarily by geographic and organizational boundaries. Cohesion and efficient operation in distributed design teams requires exceptional computational design support versus the needs of non-distributed teams [24, 30]. Information or tasks may also be distributed in the same manner as team members. When information or tasks are distributed across geographic or organizational boundaries, supplementary resources are typically required to facilitate communication. Distribution of people, information, and tasks in collaborative design often introduces challenges to efficient coordination of the design process.

4 Design Approach

Several factors of the team's design approach, shown in Table 4, may affect the collaborative design process. These factors include the design tools, how progress is evaluated, the structure of the design approach, and the process type.

Table 4. Design approach issues in collaborative design

4. Design Approach	A. Design Tools Applied in Each Phase: *Recognition of the Need, Problem Definition, Synthesis, Analysis and Optimization, Evaluation, Presentation*
	B. Evaluation of Progress: *Self-Assessment, Assessed by Outside Parties*
	C. Degree of Structure: *Company Policy, Chosen by Team, Not Well-Structured*
	D. Process Approach: *Generative, Variant*
	E. Stage: *Clarification of Task, Conceptual Design, Embodiment Design, Detail Design*

The selection of design tools (e.g., idea generation methods, decision-making techniques, risk analyses) to apply in various phases and tasks of the collaborative design process may impact the efficiency and productivity of the team. Some tools or approaches may be appropriate in team environments or particular types of problems [3]. In her analysis of the design approach at a product development firm, Parks found that the absence of a product design specification hinders task clarification activities and objective evolution of the design [31].

Studies into the effects of regular self-assessment on the performance of design teams found positive relationships between the two items [32]. This was marked by higher self-rated and group-related effectiveness when teams completed a self-assessment halfway through the design process. Group satisfaction increased when teams participated in a self-assessment.

Similarly, the degree of structure in methodology and team organization may impact the team's performance. More structure may be required for distributed teams to be productive because of the physical barriers they face [18]. In studies by Austin, et al., designers believe they have performed better as a team when they agree on and follow a design process [20]. However, no evidence was found to prove that an increase in actual productivity or success of the design team could be related to the team following a systemic design procedure. Other research, though, found that methodical design process assists in the solution development for problems in which engineers have no previous experience [31].

The primary approaches to design have been classified as generative and variant [33]. Using the variant process, the goals of the new design are achieved by adapting existing design specifications of a similar subject. Conversely, the generative process is an original design effort.

The stage of the design process also characterizes work in collaborative design. While a number of authors [2, 3, 4] have proposed definitions for the stages of the design process, the stages defined by Pahl and Beitz [4] are among the most popular and are referenced in this taxonomy. These stages include clarification of the task, conceptual design, embodiment design, and detail design.

5 Information

Information flow throughout the design process is a crucial measure of collaborative design effectiveness. Information related issues identified in the taxonomy are shown in Table 5.

Table 5. Information issues in collaborative design

5. Information	A. Form: *Design Artifact, Process Knowledge*
	B. Management: *Ownership, Permission to Change Parameters, Security, Change Propagation*
	C. Perceived Level of Criticality: *High, Medium, Low*
	D. Dependability of Information: *Reliability, Completeness*

Design information can be characterized as design artifact or process knowledge [34]. Design artifact knowledge is the actual design data and structures, such as technical charts/graphs, object attributes, and design reports, in which design data is represented. Process knowledge, or the expertise or resources that enable manipulation of design data, can be separated into reasoning (rule-based, history-based, first-principle) and tasks (search, analysis, modification).

Effective management of design information is crucial in collaborative design. The most important areas of information management in this context are ownership, permissions, security, and change management. The rights and responsibilities of various agents change throughout the design process [35]. The primary goals of managing information change are to maintain a design history, to enable backtracking, and to ensure that all agents use the most current information.

The perceived level of criticality of information is related with the selected communication method. If the information is considered highly critical, agents will likely select modes of communication in which they have the highest aptitude and confidence in reliability [23]. The quantity of communication related to a particular information exchange may also be influenced by the perceived importance of the information. The dependability of information may affect collaborative design both in terms of completeness and reliability [24]. Design information is likely to be incomplete in early stages of the design process and this level of completeness may influence the design approach. Some information may be exchanged in the design process even though the information is not yet fixed or validated. Agents in the design process should consider this reliability in decision-making and other design activities. Some options for addressing the reliability issue are probabilistic design and sensitivity analysis [4, 20].

6 Nature of Problem

Various aspects of the design problem may impact collaborative design. Primary factors in this area are shown in Table 6.

Table 6. Collaborative design issues based on the nature of the problem

6. Nature of Problem	A. Type of Design: *Novel, Routine*
	B. Coupling of Sub-Tasks: *Highly Coupled, Loosely Coupled*
	C. Level of Abstraction: *Concrete, Abstract, Intermediate*
	D. Scope: *Single Domain, Multi-Discipline*
	E. Complexity: *High, Medium, Low*

The type of design, classified as novel or routine, relates to the knowledge that is required to address a specific problem. If the design team understands what is required, then the design type may be considered routine [3]. However, if the team does not know what knowledge will be required in satisfying the design problem, the problem is classified as novel [4].

The degree of coupling of sub-tasks has a large influence on the communication requirements of the team [24]. When tasks are highly coupled, collaboration technologies with high communication impedance are acceptable. Design problems with loosely coupled tasks, though, require that obstructions to communication be at a minimum. Because the degree of coupling of tasks varies throughout the design process, resources should be available to meet the needs of highly and loosely coupled tasks. Coupling of tasks may also impact the information management requirements of the team, notably in the area of change propagation.

Design teams must typically deal with abstraction of the design problem at some level where design is a continuous process of refining the design problem to a less abstract state [3]. Some design tools and techniques are better suited for certain levels of abstraction than others. Design teams may be better equipped than individual designers to handle this ambiguity because of the range of experiences and expertise present within the group [10].

If the design problem spans a wide range of disciplines, collaborative design teams may be better equipped to handle the problem than a traditional design team [10]. As noted earlier, the ability to utilize resources from various organizations and locations may enable a team to be constructed with a desired composition of expertise. A multi-disciplinary team, however, may encounter communication and organizational challenges.