UNIT 4 -
VISIBLE-SURFACE DETERMINATION

VISIBLE SURFACES DETERMINATION

- The surface that are blocked or hidden from view must be removed in order to construct view of 3D scene.
- The identification & removing of these surfaces is called the **Hidden-surface elimination method.**
- Visible Surface Detection is a key step in the process of the 3D viewing pipeline along with **rendering** or **illumination**.
- Rendering - It refers to the process of adding realism to computer graphics by adding 3D qualities such as shadows and variations in color & shade.
- Illumination – Described by an equation to model the interaction of light with the surface of some objects.
- Visible Surface Detection or Visible Surface Determinations.

CATEGORIES OF ALGORITHMS

1) Object space method - called as continuous domain, continuous operation which **compares parts of objects** to each other to determine which surfaces should be levelled as visible.

2) Image space methods - the visibility is decided point by point at each pixel on the **projection plane** and this screened resolution can be a limitation in this particular case

SORTING & COHERENCE

- Both the methods are used to improve performance.
- Sorting is used to facilitate depth comparisons by ordering the individual surfaces in the scene according to their distance from the view plane.
- Coherence is used to take an advantage of regularities in a scene.
  - Scan line intervals & patterns
  - Animation frames in moving objects.

COHERENCE FOR VISIBILITY

- Coherence is a term used to represent similarities between objects.
- It is based on the principle of locality where nearby objects have similar characteristics.
- It is used for hidden surface removal.

TYPES OF COHERENCE

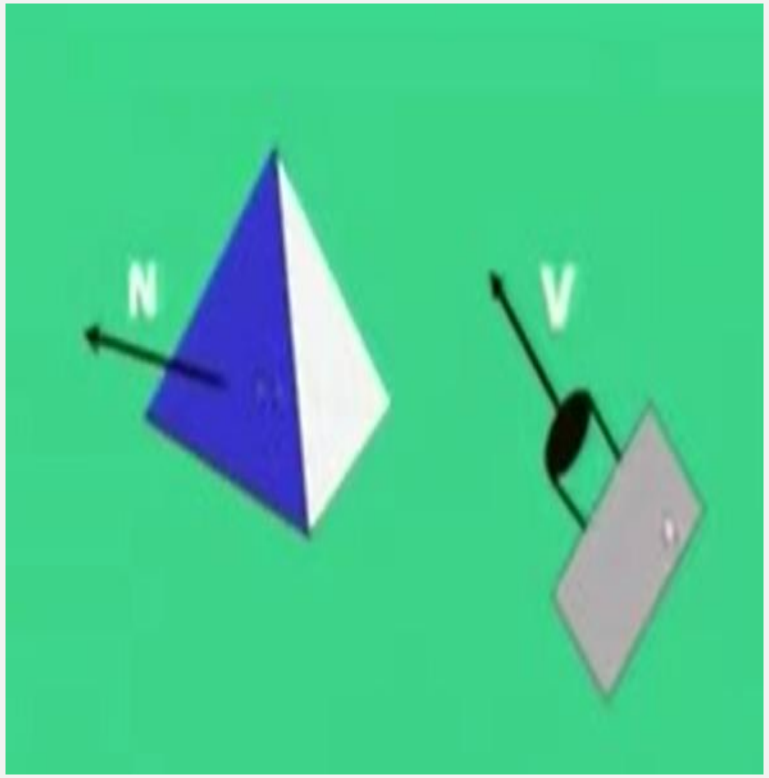
- **Object coherence -** if polygons are totally different don't compare.
- **Face coherence -** smooth variation across face, incrementally modify.
- **Edge coherence -** visibility changes if edge crosses behind a visible surface.
- **Implied Edge coherence -** Line of intersection of a planner face penetrating another.
- **Scan line coherence -** successive line have similar spans.
- **Depth coherence -** use difference equations to estimate depth of nearby points on the same surface.
- **Area coherence -** span of adjacent group of pixels is often covered by the same visible face.
- **Frame coherence -** pictures of two successive frames of an animation sequence are quite similar.

VISIBLE SURFACE DETECTION METHOD

- 1) Back face removal
- 2) The Z-Buffer Algorithm
- 3) Scan-line method
- 4) Painter’s algorithms (depth sorting)
- 5) Area sub-division method
- 6) BSP trees
- 7) Visible-Surface Ray Tracing

1) BACK FACE REMOVAL

- Object surfaces that are oriented away from the viewer are called back faces.
- The back faces of a cube are completely blocked by the cube itself and hidden from view.
- We can identify and remove these back faces by equations.
- V- View Vector, N- Surface normal
- v.n>0
- The equation of plane
- Ax+By+Cz+D=0
- If Ax+By+Cz+D>=0 surface is back face.



2) THE Z-BUFFER ALGORITHM

- Also called as Depth Buffer algorithm.
- This algorithm compares depth at each pixel position on the projection plane.
- The surface depth is measured from the view plane along the z-axis of the view system.
- The implementation requires another buffer memory called Z-buffer along with the frame buffer memory required for raster display devices.
- A Z-buffer is used to store depth values for each (x, y) position as surfaces are processed, and the frame buffer stores the intensity values for each position.
- At the beginning Z-buffer is initialized to zero.
- The frame buffer is initialized to the background colour.

Z-BUFFER ALGORITHM

1. Initialize the Z-buffer and frame buffer so that for all buffer positions

   z-buffer(x,y)=0 and Frame-buffer(x,y)=$I_{background}$

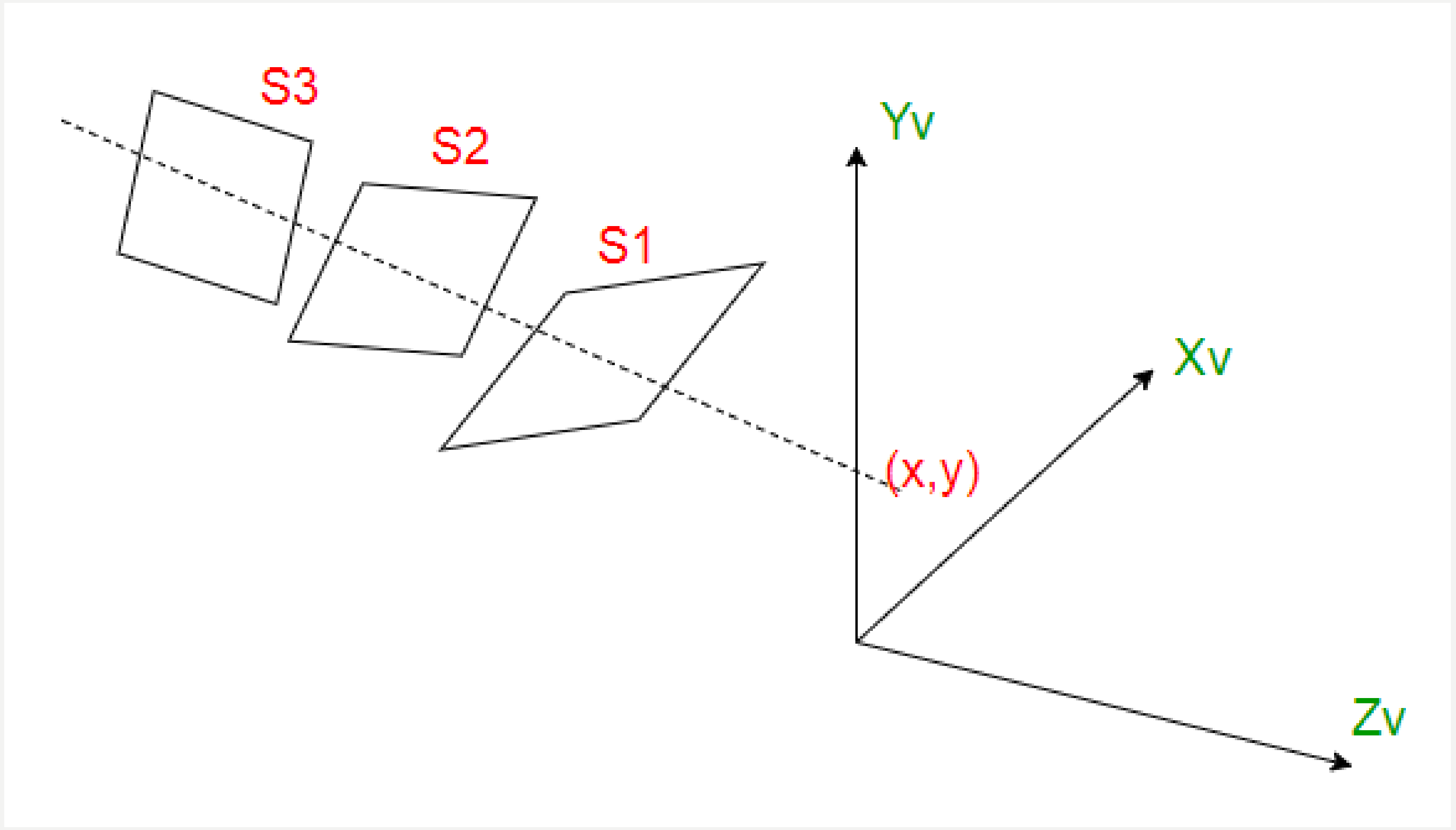
2. During scan conversion process, for each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
3. Calculate z-value for each (x,y) position on the polygon

```
If z>z-buffer(x,y) then
    z-buffer(x,y)=z,
Frame-buffer(x,y)= $I_{surface}$(x,y).
```

4. Stop.

(After processing of all surfaces, the z-buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding intensity values for those surfaces.)

Z-BUFFER ALGORITHM



Advantages

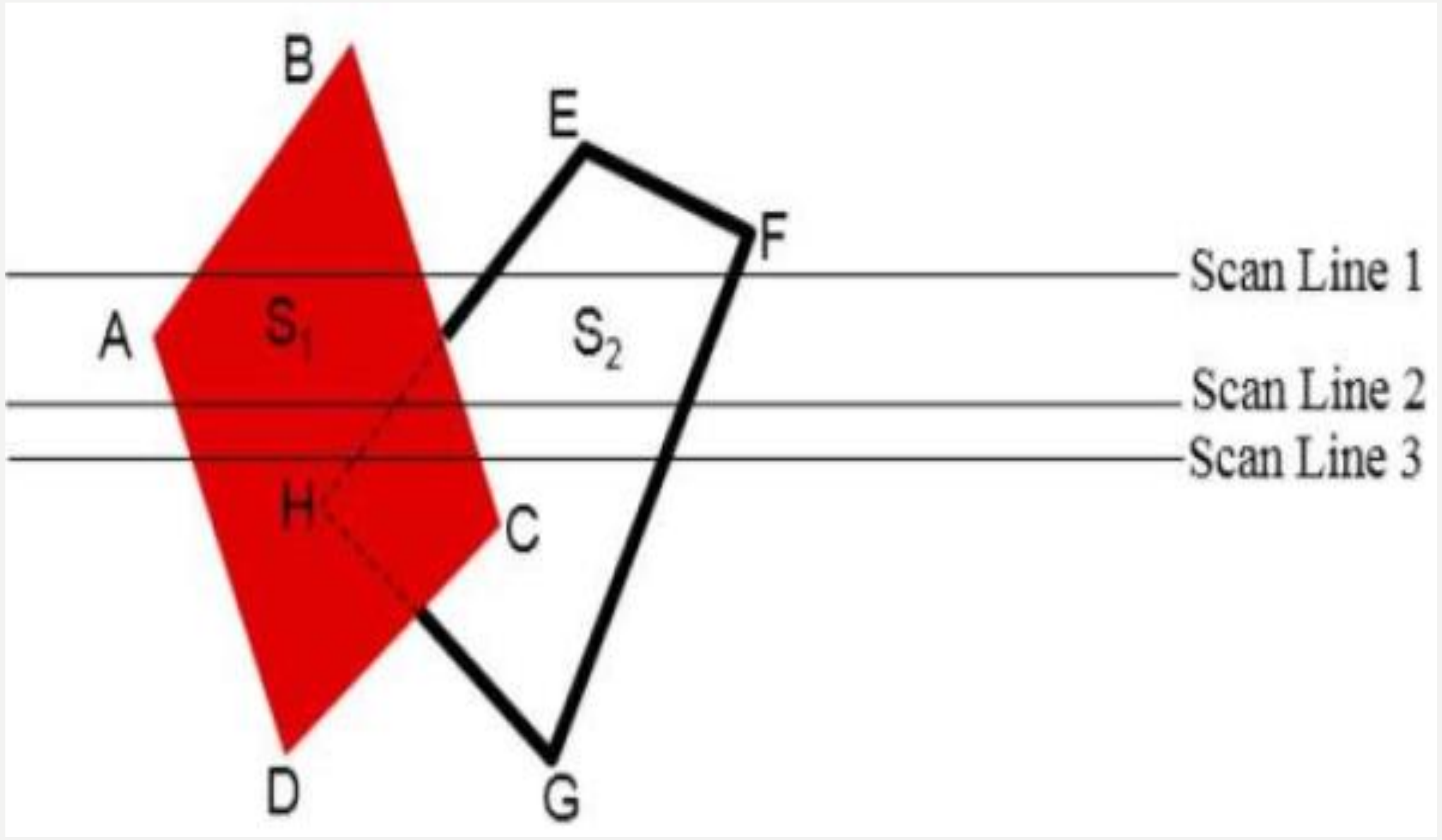
1. It is easy to implement.
2. It can be implemented in hardware to overcome the speed problem.
3. Since the algorithm processes objects one at a time, the total number of polygons in a picture can be arbitrarily large.

Disadvantages

1. It requires an additional buffer and hence the large memory.
2. It is time consuming process as it requires comparison for each pixel instead of for the entire polygon.

3) SCAN-LINE METHOD

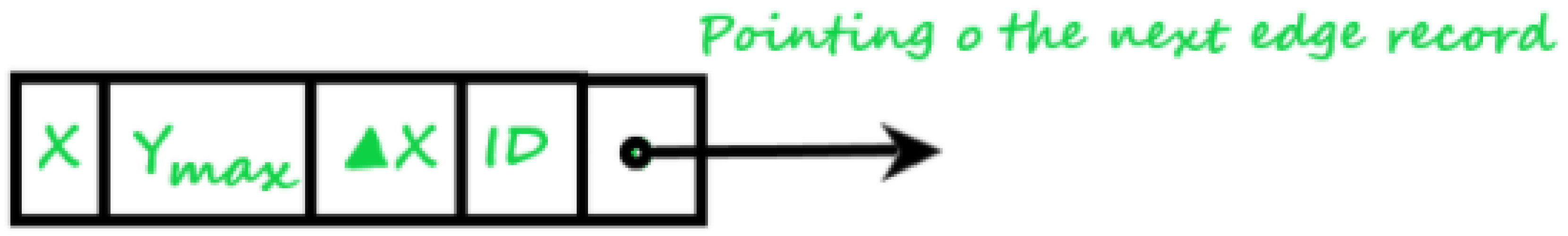
- It is an image-space method to identify visible surface. This method has a depth information for only single scan-line. In order to require one scan-line of depth values, we must group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line. Two important tables, **edge table** and **polygon table,** are maintained for this.
- **The Edge Table** − It contains coordinate endpoints of each line in the scene, the inverse slope of each line, and pointers into the polygon table to connect edges to surfaces.
- **The Polygon Table** − It contains the plane coefficients, surface material properties, other surface data, and may be pointers to the edge table.


B
E
F
Scan Line 1
A
S_1
S_2
Scan Line 2
Scan Line 3
H
C
D
G

- To facilitate the search for surfaces crossing a given scan-line, an active list of edges is formed. The active list stores only those edges that cross the scan-line in order of increasing x. Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface.
- Pixel positions across each scan-line are processed from left to right. At the left intersection with a surface, the surface flag is turned on and at the right, the flag is turned off. You only need to perform depth calculations when multiple surfaces have their flags turned on at a certain scan-line position.
- Moving across a scan line the flag for a polygon is flipped when an edge of that polygon is crossed.
- If no flags are true then nothing is drawn
- If one flag is true then the colour of that polygon is used
- If more than one flag is true then the front most polygon must be determined.

EDGE LIST TABLE

1. Edges list table(list): This list maintains the record of all the edges by storing their endpoint coordinates. The x-coordinate that we choose, whose Y-coordinate = $Y_{min.}$

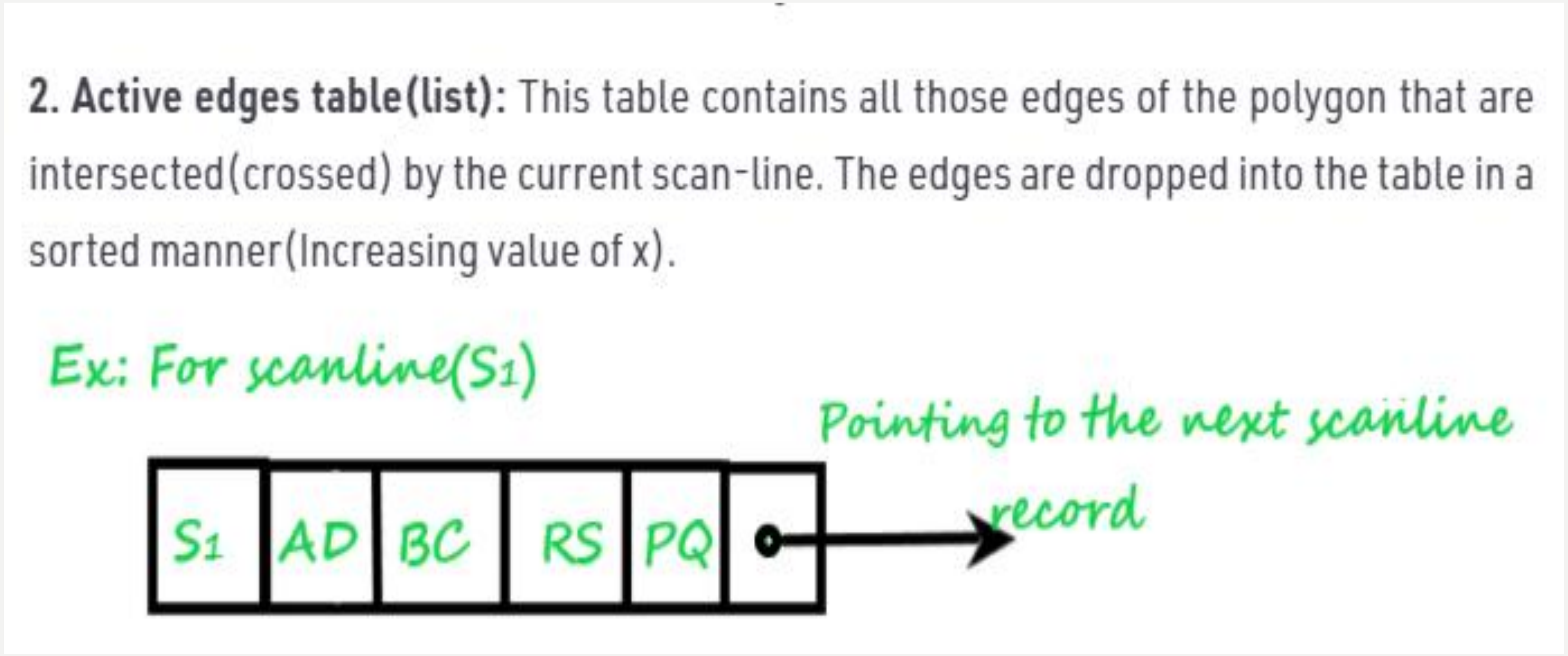


$\blacktriangle X=\blacktriangle Y/m= (Y_{max}-Y_{min})/m= 1/m$,where, m=slope and

$\blacktriangle Y=Y_{max}-Y_{min} \cong 1/m$,where Y_{min} and Y_{max} are the two end pointsY-coordinate of the edge

ID: Corresponding ID of the edge that has been stored

ACTIVE EDGE TABLE

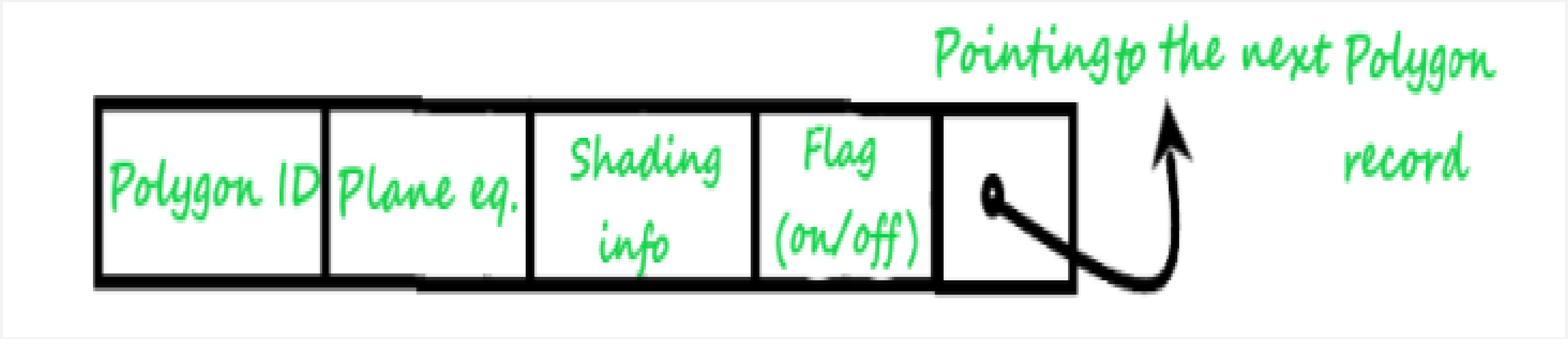


2. Active edges table(list): This table contains all those edges of the polygon that are intersected(crossed) by the current scan-line. The edges are dropped into the table in a sorted manner(Increasing value of x).

POLYGON TABLE

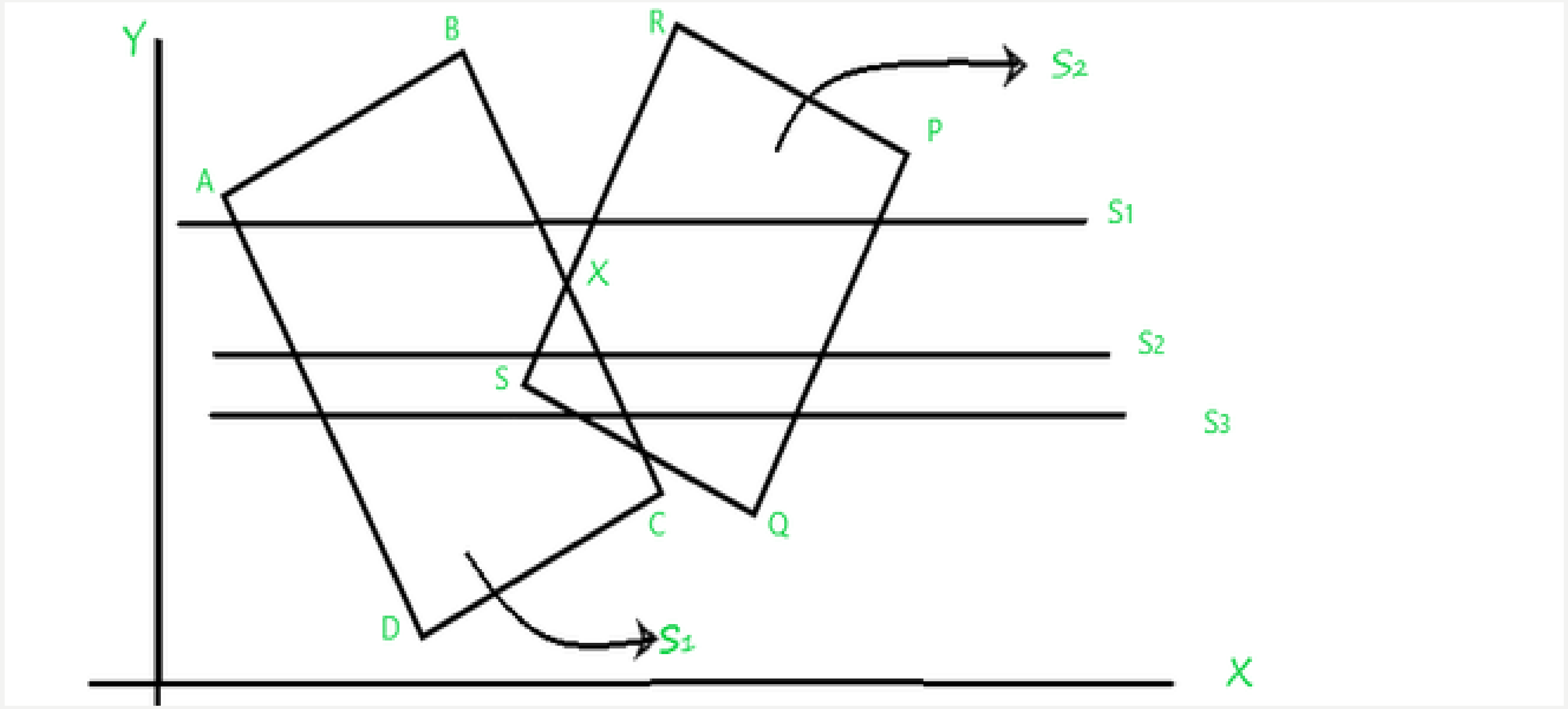
3. Polygon table(list): This list consists of:

- Polygon Id.
- Plane equation.
- Color Information of the surface.
- Flag of surface(on/off)].



EXAMPLE

- Here, two overlapped polygons are given which are intersected by three Scan-lines S_1, S_2, S_3 respectively. So, What happens if the Scan-line algorithm is applied in order to identify the Hidden surface(visible surface).

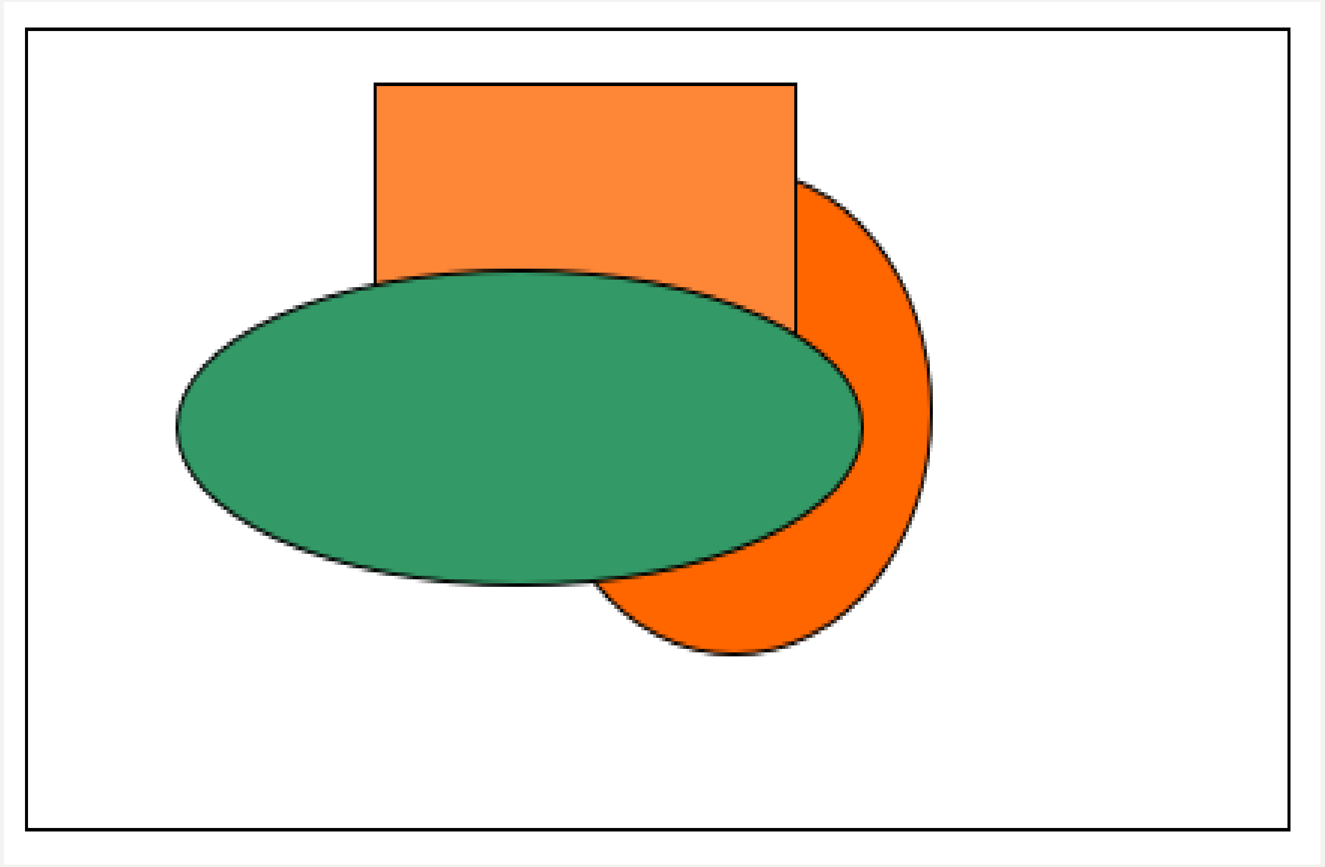


- **1. First, examine the scanline(S_1), whose,**
- Active edge table (Aet) contains: [AD,BC,RS,PQ], and
- The flag is set to “on” for surface(S_1) and surface(S_2) and the flag set “off” for the region between BX and RX as it’s an outer region of the polygon’s surface and not to be projected at view-port(display devices), now
- Drop the color-intensities of the corresponding surfaces into the frame buffer(refresh buffer).
- **2. Then, process the scanline(S_2), whose,**
- Active edge table (Aet) contains: [AD,BC,RS,PQ], and
- The flag is set to “on” for surface(ABCD) and surface(PQRS),
- Both of the polygons surfaces are overlapping each other so for this overlapped region which of the surface intensity should be taken into account. So to answer this calculates the depth(Z_{min}) of both surface(S_1) and surface(S_2) of this overlapped portion, next,
- If depth(S_1)>depth(S_2), then the Flag of surface S_1=” on” and intensity of surface S_1 will be considered else S_2, now
- Drop the color-intensities of the corresponding surfaces whose flag is set to on into the frame buffer(refresh buffer).

- **3.** As Scanline(S_3) is passing through the same portion from where Scanline(S_2) is passing, S_3 also has the same Active edge table(Aet) components as S_2 has and no need to calculate the depth(S1) and depth(S2) again so S_3 can take the advantage of the concept of Coherence.

4) PAINTER'S ALGORITHMS (DEPTH SORTING)

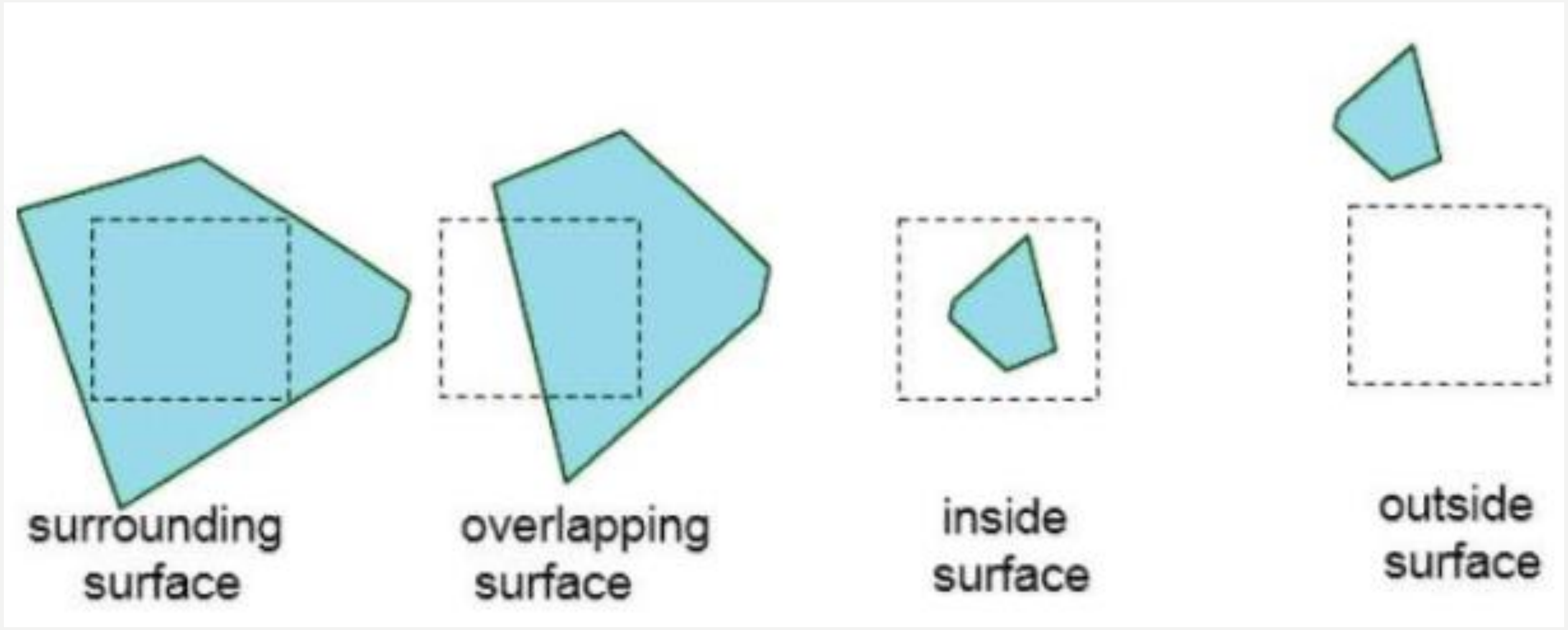
- The idea behind painter's algorithm is to paint the polygon into frame buffer in order of decreasing distance.
- The algorithm gets its name from the manner in which an oil painting is created. The artist begins with the background.
- He then adds the most distant object and then the newer object and so on.
- There is no need to erase the portion of background, the artist simply paints on top of them.
- Whole canvas is painted first then on top this canvas a circle is painted
- On top this circle a rectangle is painted.
- On top of this rectangle an ellipse is painted.
- It gives an appearance as the ellipse is nearer to the viewer then the circle.
- The main factor here is to check the priority
- Each polygon is assigned a priority number
- Polygons are sorted according to their priority.
- Lowest one first, followed by higher then highest.



5) AREA SUB-DIVISION METHOD

- The area-subdivision method takes advantage by locating those view areas that represent part of a single surface.
- Divide the total viewing area into smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.
- Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.
- An easy way to do this is to successively divide the area into four equal parts at each step.
- There are four possible relationships that a surface can have with a specified area boundary.

- **Surrounding surface** − One that completely encloses the area.
- **Overlapping surface** − One that is partly inside and partly outside the area.
- **Inside surface** − One that is completely inside the area.
- **Outside surface** − One that is completely outside the area.



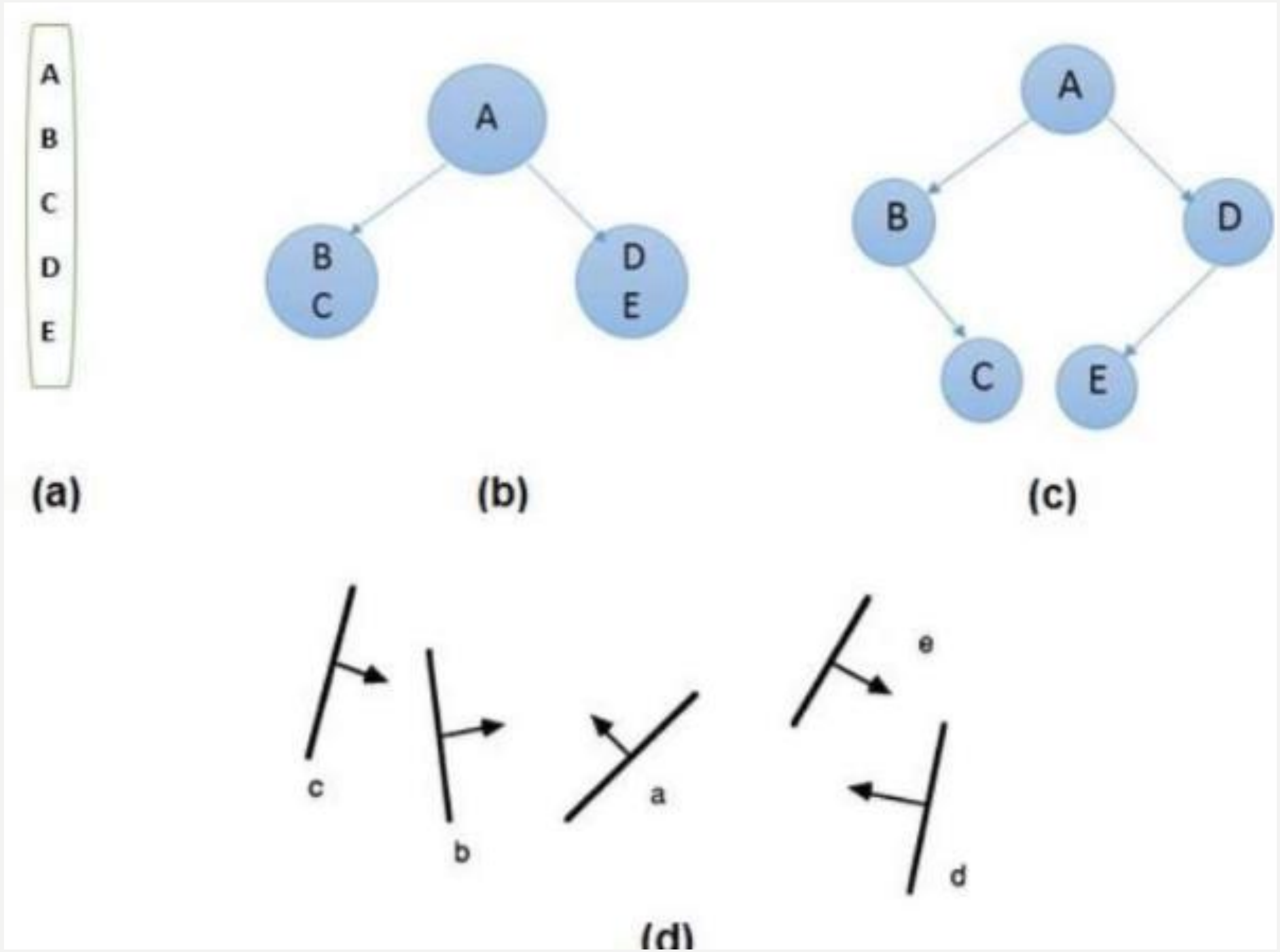
- The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true −
- All surfaces are outside surfaces with respect to the area.
- Only one inside, overlapping or surrounding surface is in the area.
- A surrounding surface obscures all other surfaces within the area boundaries.

1. If there is a single surrounding polygon and no intersecting or contained polygons then the area is filled with the colour of the surrounding polygon.
2. If there is a single contained polygon or intersecting polygon then the background colour is used to fill the area, then the part of the polygon contained in the area is filled with the colour of that polygon
3. If there is a surrounding polygon in front of any other surrounding, intersecting, or contained polygons then the area is filled with the colour of the front surrounding polygon
4. If all polygons are disjoint then the background colour fills the area

6) BSP TREES

- Binary space partitioning is used to calculate visibility. To build the BSP trees, one should start with polygons and label all the edges.
- Dealing with only one edge at a time, extend each edge so that it splits the plane in two.
- Place the first edge in the tree as root.
- Add subsequent edges based on whether they are inside or outside.
- Edges that span the extension of an edge that is already in the tree are split into two and both are added to the tree.

- It is a depth sort algorithm with a large amount of pre-processing to create a data structure to hold the polygons.
- First generate a 3D BSP tree for all of the polygons in the scene
- Then display the polygons according to their order in the scene
- Polygons behind the current node
- Polygons in front of the current node

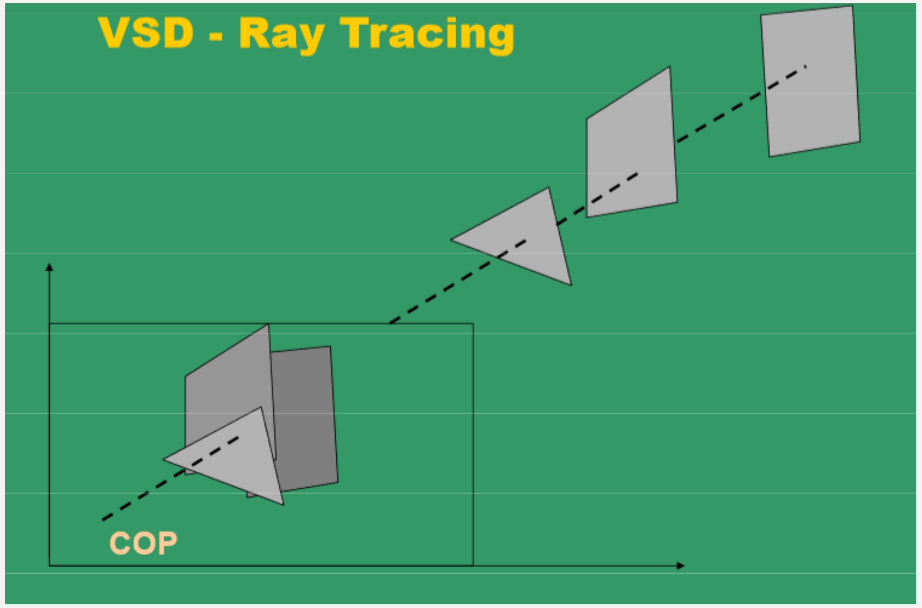

A
B
C
D
E
(a)
A
B C
D E
(b)
A
B
D
C
E
(c)
c
b
a
e
d
(d)

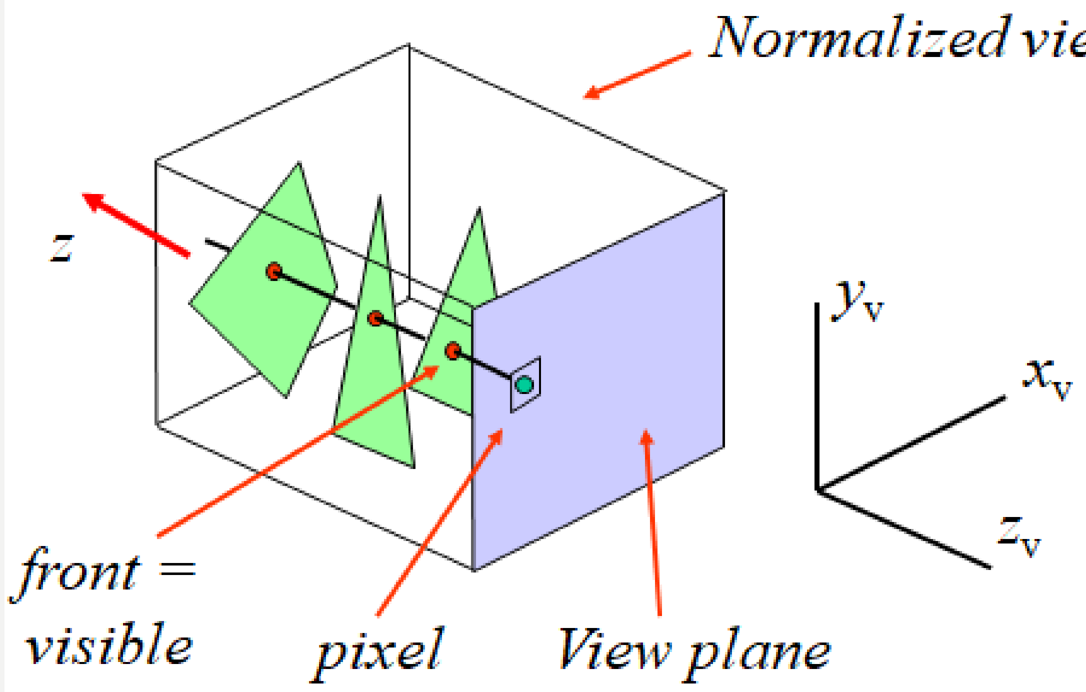
- From the above figure, first take A as a root.
- Make a list of all nodes in figure (a).
- Put all the nodes that are in front of root A to the left side of node A and put all those nodes that are behind the root A to the right side as shown in figure (b).
- Process all the front nodes first and then the nodes at the back.
- As shown in figure (c), we will first process the node B. As there is nothing in front of the node B, we have put NIL. However, we have node C at back of node B, so node C will go to the right side of node B.
- Repeat the same process for the node D.

- Each node in the tree is a polygon. Extending that polygon generates a plane. That plane cuts space into 2 parts. We use the front-facing normal of the polygon to define the half of the space that is 'in front' of the polygon. Each node has two children: the front children (the polygons in front of this node) and the back children (the polgons behind this noce)
- In doing this we may need to split some polygons into two.
- Then when we are drawing the polygons we first see if the viewpoint is in front of or behind the root node. Based on this we know which child to deal with first - we first draw the subtree that is further from the viewpoint, then the root node, then the subtree that is in front of the root node, recursively, until we have drawn all the polygons.
- Compared to depth sort it takes more time to setup but less time to iterate through since there are no special cases.

7) VISIBLE-SURFACE RAY TRACING

- Ray tracing is an image based algorithm. For every pixel in the image, a ray is cast from the center of projection through that pixel and into the scene. The colour of the pixel is set to the colour of the object that is first encountered.





Algorithm:
Per pixel:
- Calculate intersection points
- Determine front one

H&B 16-10:518-51

COMPARISON OF THE METHODS

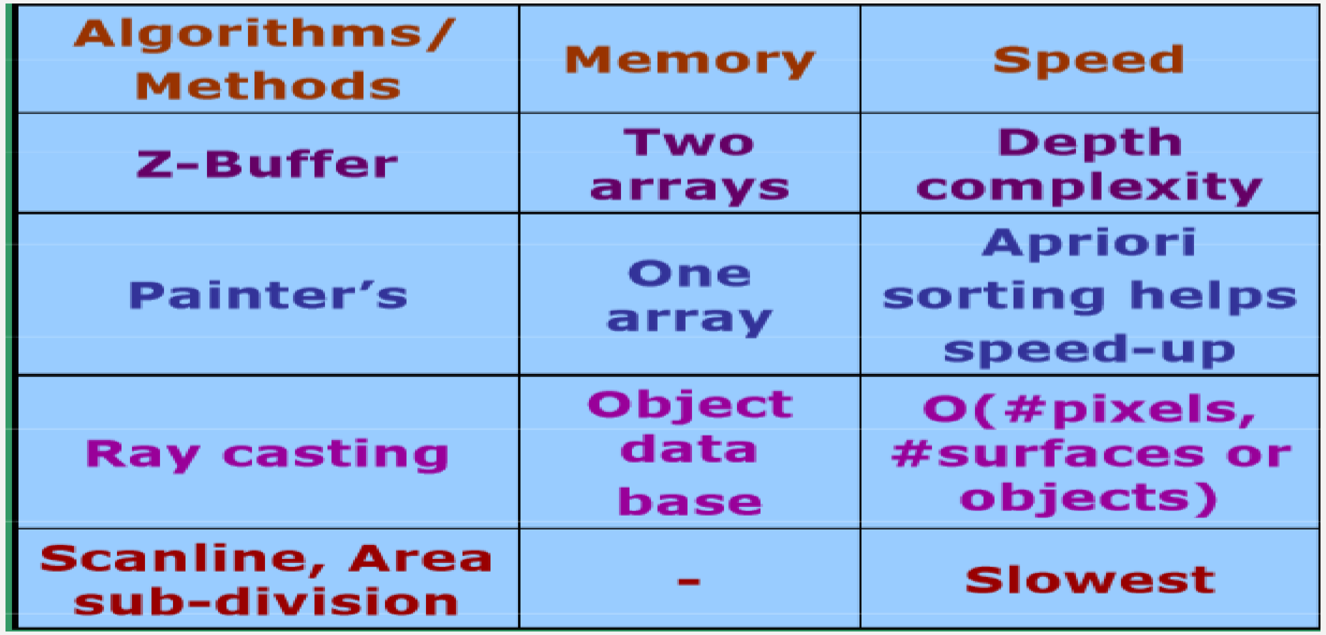
Hardware available?

Use depth-buffer, possibly in combination with back-face elimination or depth-sort for part of scene.

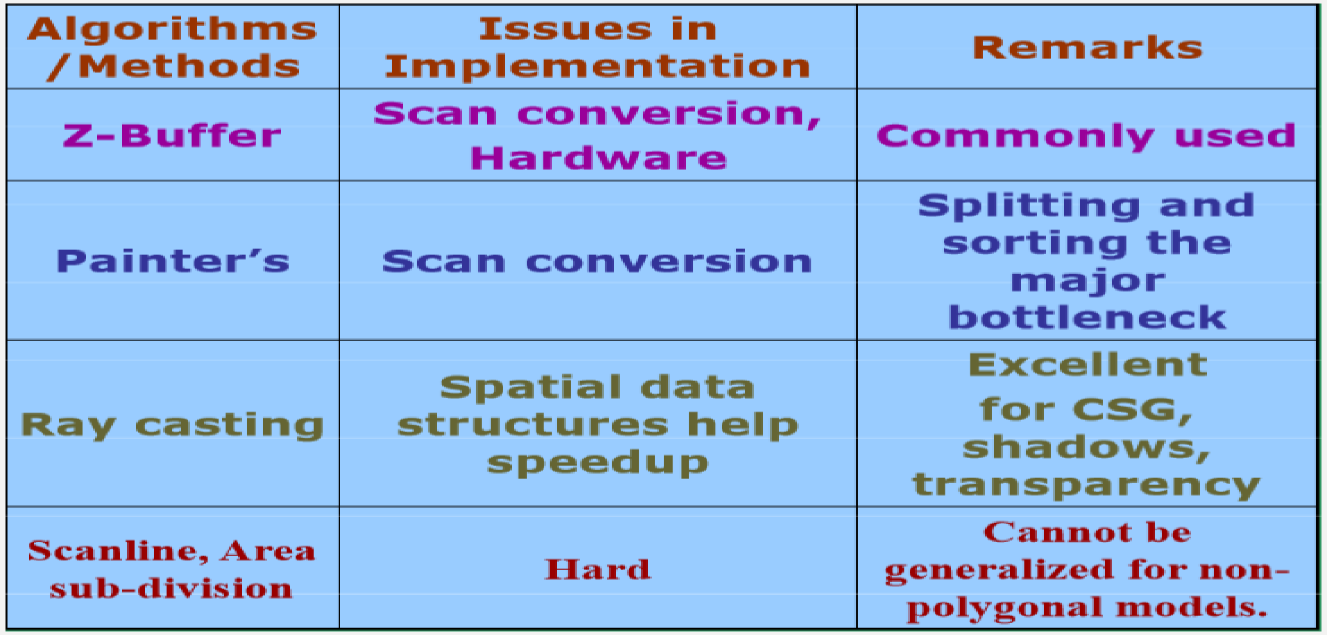
If not, choose dependent on complexity scene and type of objects:

- Simple scene, few objects: depth-sort
- Quadratic surfaces: ray-casting
- Otherwise: depth-buffer

Many additional methods to boost performance (kD-trees, scene decomposition, etc.)



Algorithms/ Methods	Memory	Speed
Z-Buffer	Two arrays	Depth complexity
Painter’s	One array	Apriori sorting helps speed-up
Ray casting	Object data base	O(#pixels, #surfaces or objects)
Scanline, Area sub-division	-	Slowest



Algorithms /Methods	Issues in Implementation	Remarks
Z-Buffer	Scan conversion, Hardware	Commonly used
Painter’s	Scan conversion	Splitting and sorting the major bottleneck
Ray casting	Spatial data structures help speedup	Excellent for CSG, shadows, transparency
Scanline, Area sub-division	Hard	Cannot be generalized for non-polygonal models.

CURVES

CURVE REPRESENTATION

- Curves can be described mathematically by **nonparametric** or **parametric** equations.
- Nonparametric equations can be explicit or implicit.
- For a nonparametric curve, the coordinates y and z of a point on the curve are expressed as two separate functions of the third coordinate x as the independent variable.
- This curve representation is known as the nonparametric explicit form.
- If the coordinates x, y and z are related together by two functions, a nonparametric implicit form results.

- There are three problems with describing curves using nonparametric equations:

(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.

(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.

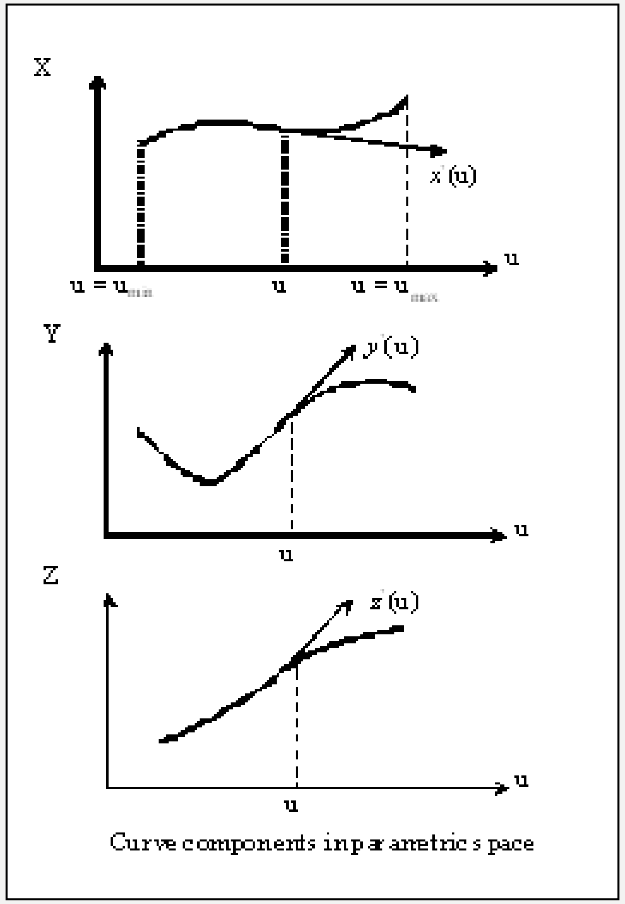
(3) If the curve is to be displayed as a series of point or straight-line segments, the computations involved could be extensive.

- Parametric representation allows closed and multiple-valued functions to be easily defined and replaces the use of slopes with that of tangent vectors, as will be introduced shortly.
- In parametric form, each point on a curve is expressed as a function of a parameter u.
- The parametric equation for a

three-dimensional curve in space takes the

following vector form:

- $P[u] = [x\ y\ z]T$ where umax<= u <= umin



PARAMETRIC REPRESENTATION OF A CIRCLE

The parametric equations of a circle centered at the origin with radius r,

$$x = r\cos t$$

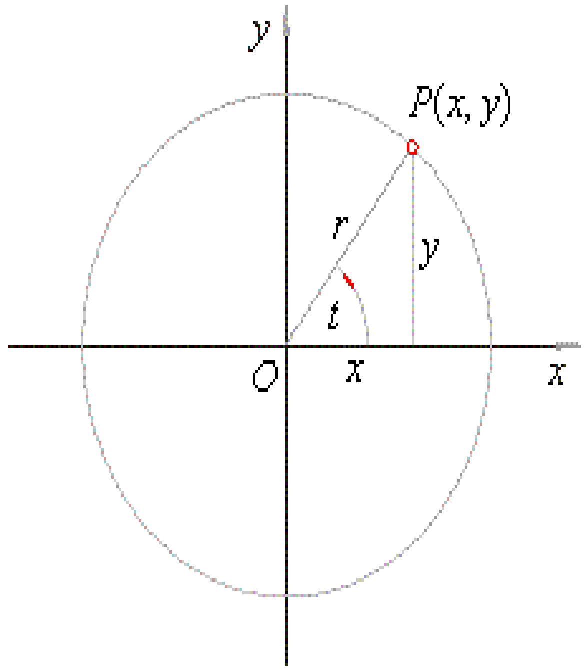
$$y = r\sin t$$

where, $0 \leq t < 2\pi$.

To convert the above equations into Cartesian coordinates, square and add both equations, so we get

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

as $\sin^2 t + \cos^2 t = 1$.



The parametric equations of a circle with center (x_0, y_0) and radius r,

$$x = x_0 + r\cos t$$
$$y = y_0 + r\sin t$$

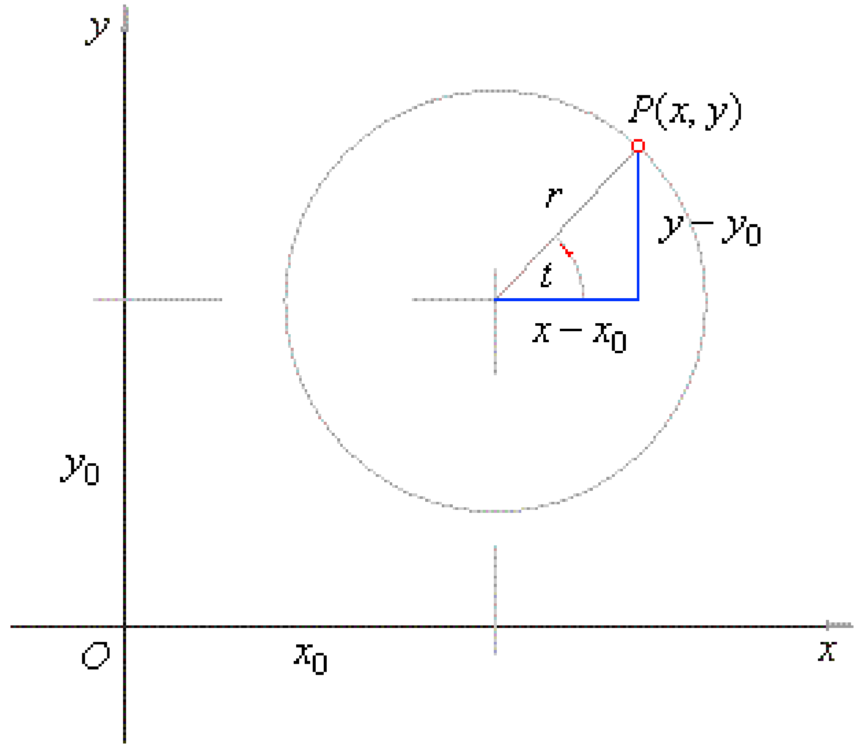
where, $0 \leq t < 2\pi$.

If we write the above equations,

$$x - x_0 = r\cos t$$
$$y - y_0 = r\sin t$$

then square and add them we get the equation of the translated circle in Cartesian coordinates,

$$(x - x_0)^2 + (y - y_0)^2 = r^2.$$



- Parametrically
- y(t)=rsin(2.pi.t)
- x(t)=r.cos(2.pi.t)
- When t runs from 0 to 1 the angle argument runs a full circle and point (x(t),y(t)) describes circle.

PARAMETRIC REPRESENTATION OF ELLIPSE

The parametric equations of an ellipse centered at the origin

Recall the construction of a point of an ellipse using two concentric circles of radii equal to lengths of the semi-axes a and b, with the center at the origin as shows the figure, then

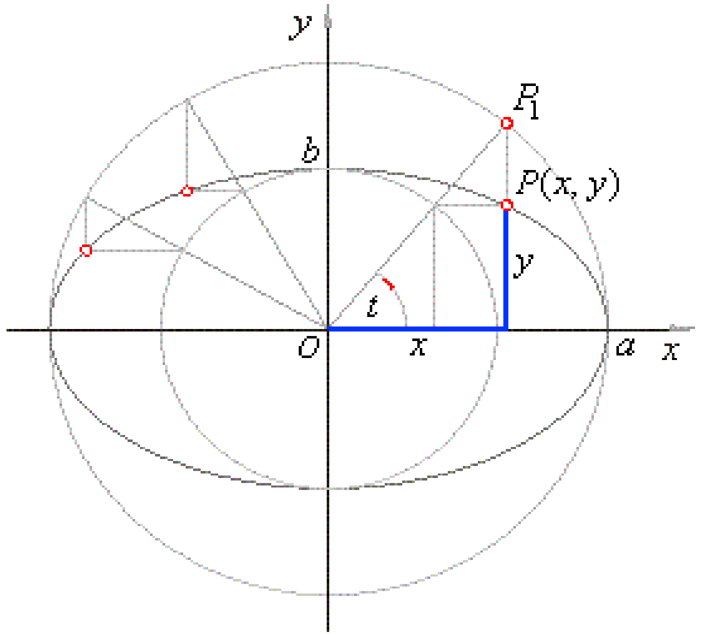
$$x = a \cos t$$
$$y = b \sin t$$

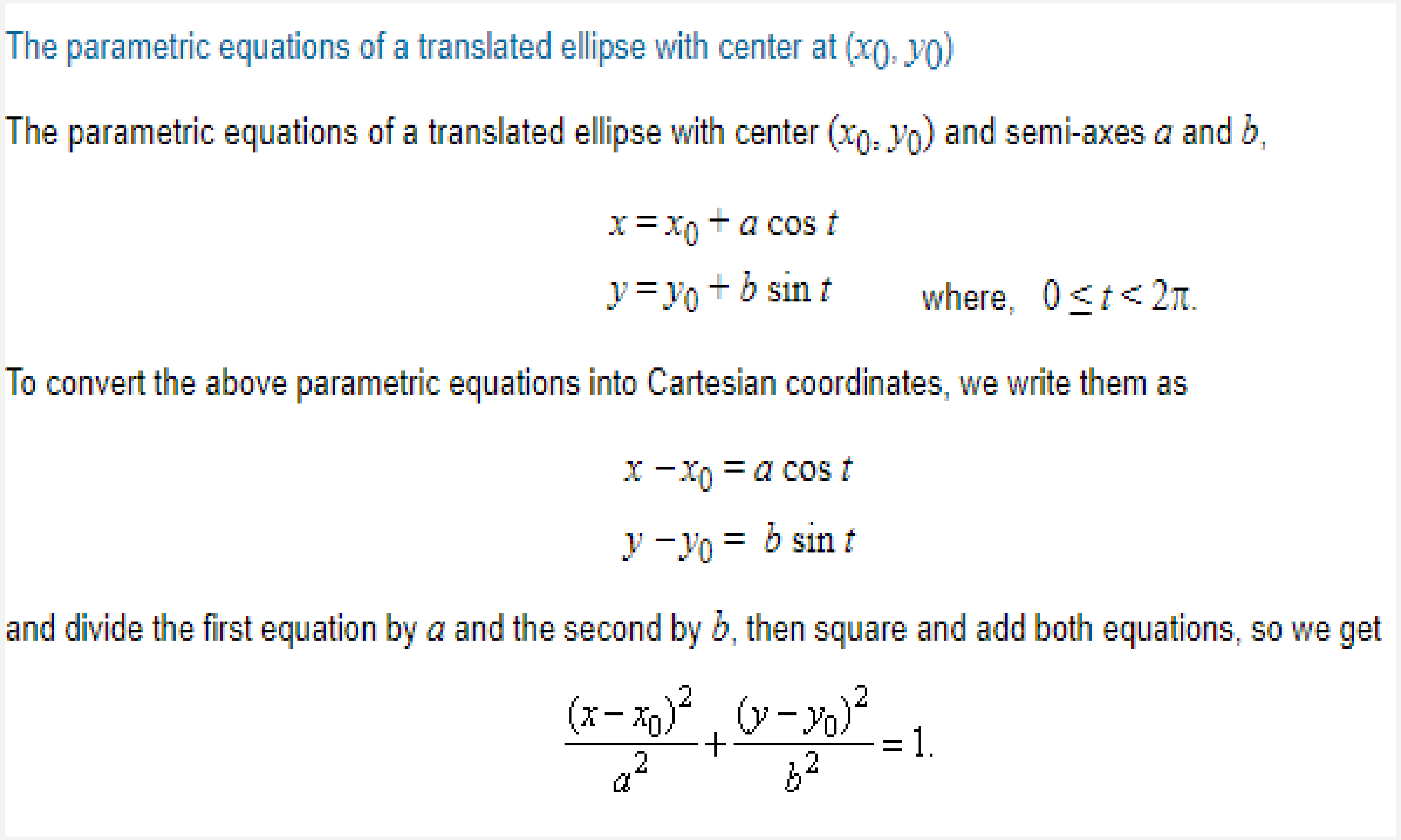
where, $0 \leq t < 2\pi$.

To convert the above parametric equations into Cartesian coordinates, divide the first equation by a and the second by b, then square and add them,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

thus, obtained is the standard equation of the ellipse.





The parametric equations of a translated ellipse with center at (x_0, y_0)

The parametric equations of a translated ellipse with center (x_0, y_0) and semi-axes a and b,

$$x = x_0 + a \cos t$$

$$y = y_0 + b \sin t \qquad \text{where,} \quad 0 \leq t < 2\pi.$$

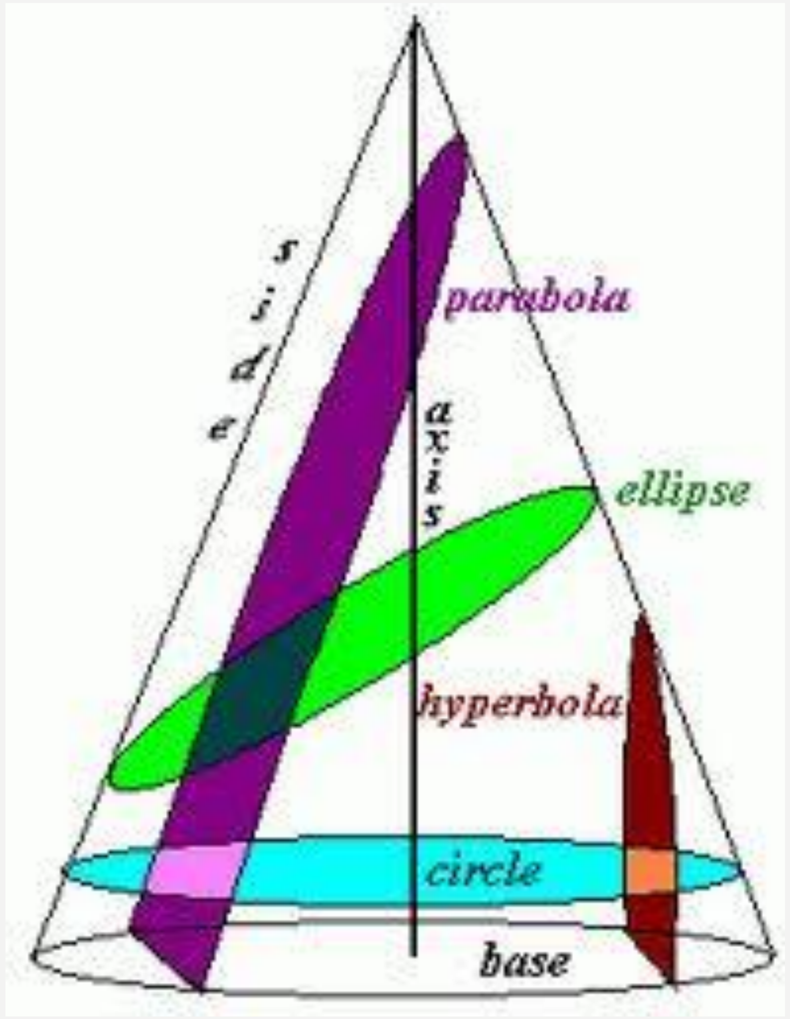
To convert the above parametric equations into Cartesian coordinates, we write them as

$$x - x_0 = a \cos t$$

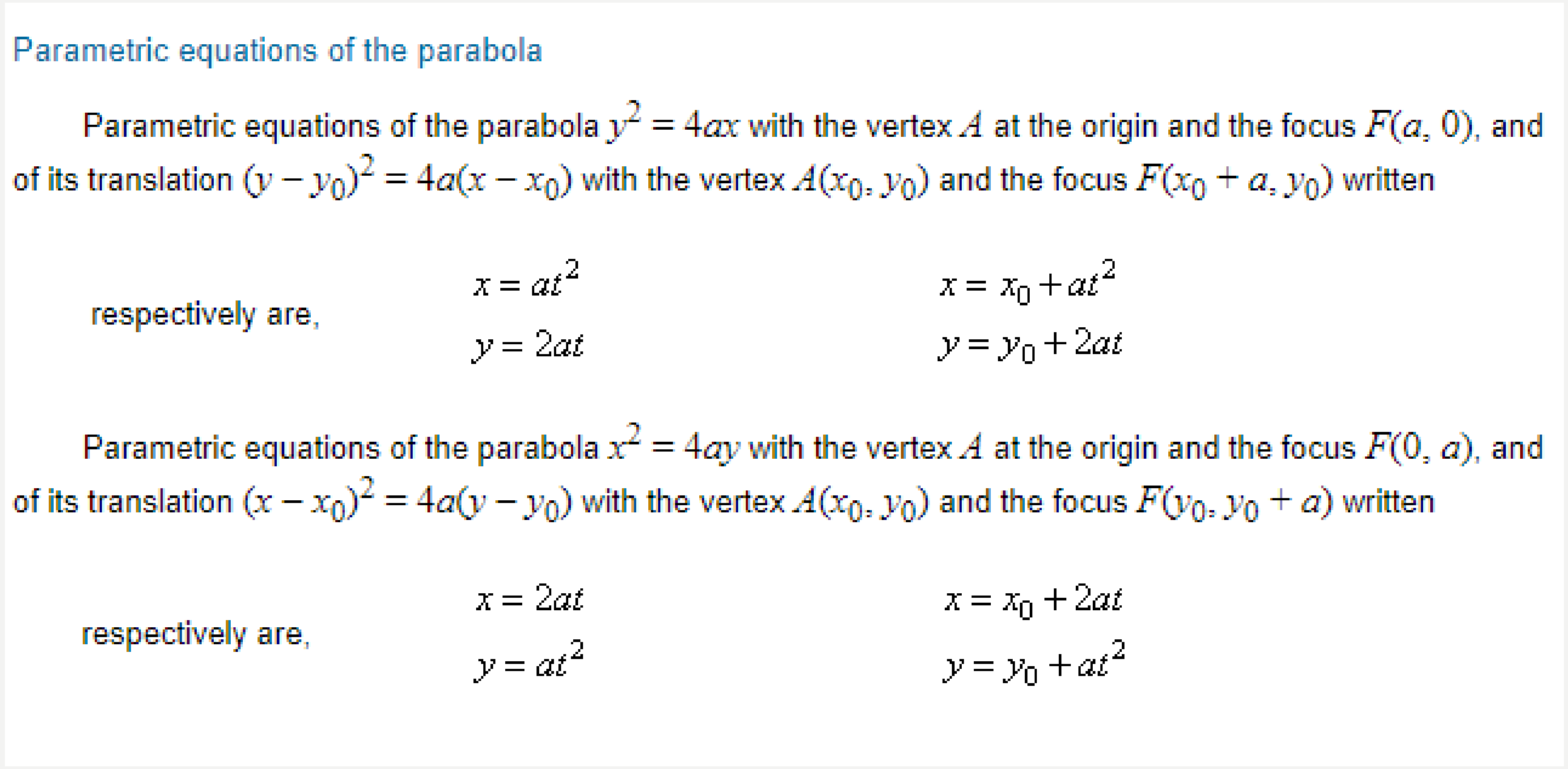
$$y - y_0 = b \sin t$$

and divide the first equation by a and the second by b, then square and add both equations, so we get

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1.$$


side
parabola
axis
ellipse
hyperbola
circle
base

PARAMETRIC REPRESENTATION OF PARABOLA



Parametric equations of the parabola

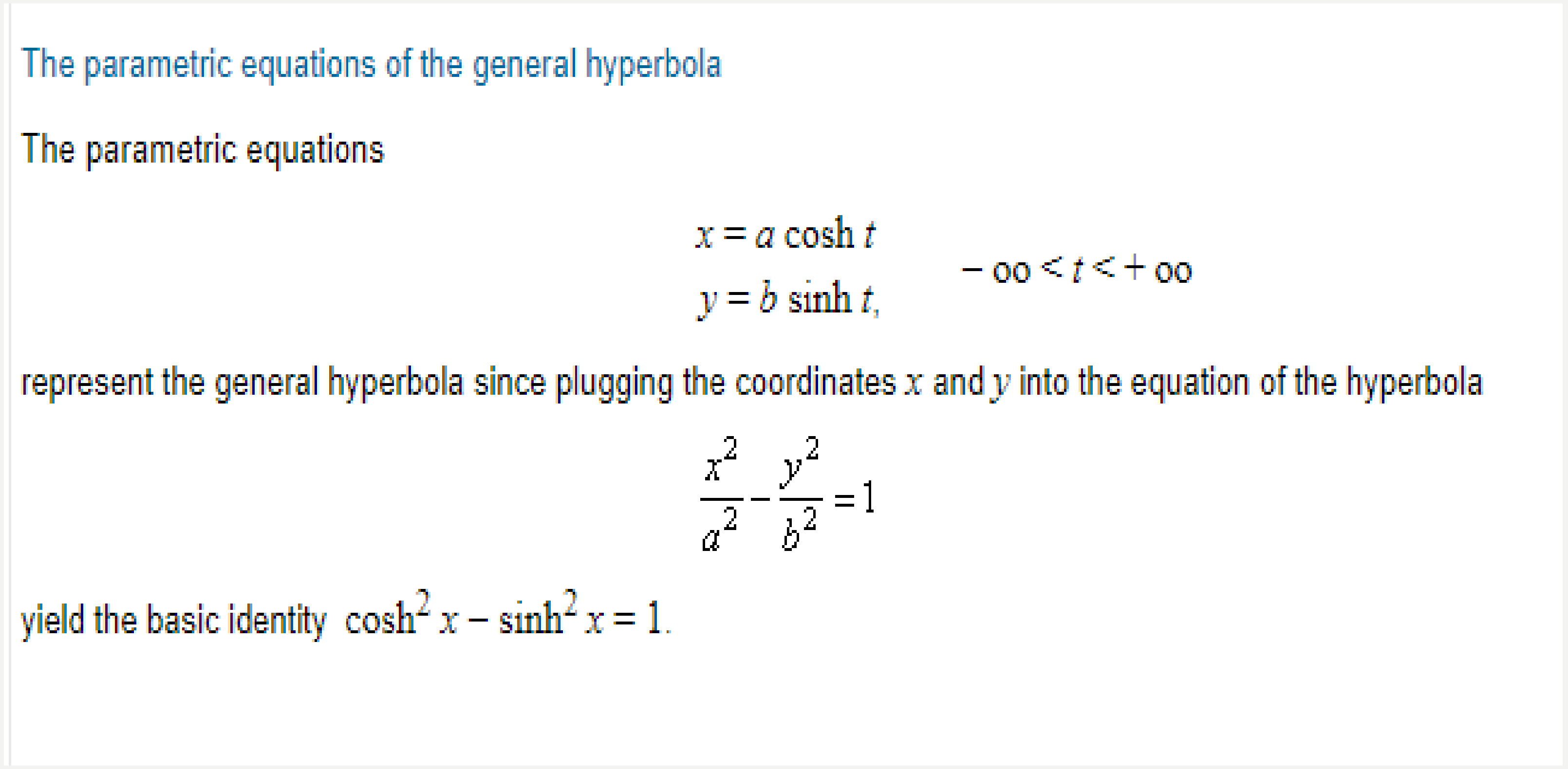
Parametric equations of the parabola $y^2 = 4ax$ with the vertex A at the origin and the focus $F(a, 0)$, and of its translation $(y - y_0)^2 = 4a(x - x_0)$ with the vertex $A(x_0, y_0)$ and the focus $F(x_0 + a, y_0)$ written

respectively are, $\quad x = at^2,\ y = 2at \quad\quad x = x_0 + at^2,\ y = y_0 + 2at$

Parametric equations of the parabola $x^2 = 4ay$ with the vertex A at the origin and the focus $F(0, a)$, and of its translation $(x - x_0)^2 = 4a(y - y_0)$ with the vertex $A(x_0, y_0)$ and the focus $F(y_0, y_0 + a)$ written

respectively are, $\quad x = 2at,\ y = at^2 \quad\quad x = x_0 + 2at,\ y = y_0 + at^2$

PARAMETRIC REPRESENTATION OF HYPERBOLA



The parametric equations of the general hyperbola

The parametric equations

$$x = a \cosh t \qquad -\infty < t < +\infty$$
$$y = b \sinh t,$$

represent the general hyperbola since plugging the coordinates x and y into the equation of the hyperbola

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

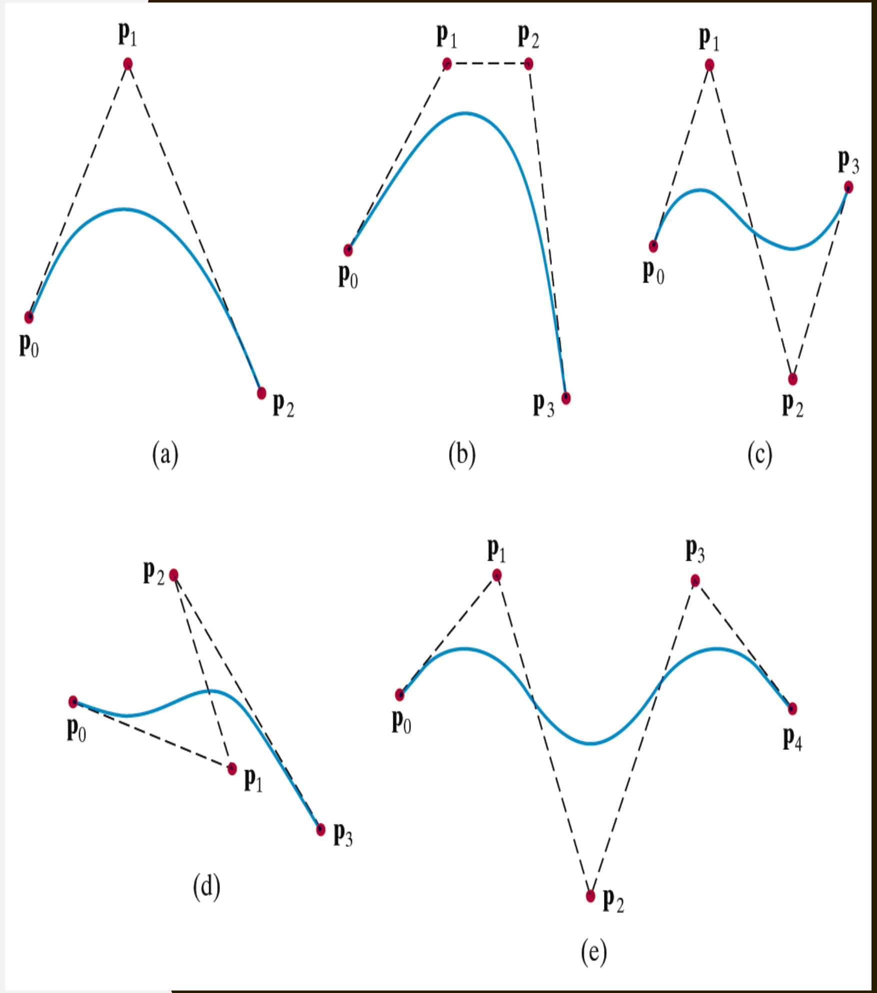
yield the basic identity $\cosh^2 x - \sinh^2 x = 1$.

- A hyperbola is like a ellipse turned inside out. For any point on the hyperbola the difference between the distances to the foci is a constant . Hyper bola consist of two curved paths, each going for ever. Two parametric forms are
- Horizontal hyperbola
- F(t)=(x(t),y(t))
- x(t)=asec(t)
- y(t)=btan(t)
- Vertical hyperbola
- F(t)=(x(t),y(t))
- x(t)=btan(t)
- y(t)= asec(t)

BEZIER CURVES

- It was developed by the French engineer Pierre Bezier for use of the design in Renault automobile bodies.
- They are highly useful & convenient for curve & surface design.
- Easy o implement.
- For these reasons, Bezier curves or splines are widely used in various CAD Systems, in general graphics packages & in sorted drawings & paintings packages.

- Bezier curves must have precisely n control points, where n+1 is the degree of the Bezier polynomial.
- By connecting Bezier curves we can create long curves.
- This is done by connecting last control point of one curve the same as first control point of next curve.
- This type of curve always lie within the convex hull of the control points & always have the sum of the basis functions add to 1.
- A convex hull of set of points is the smallest convex polygon that contains one of the points.



PROPERTIES OF BEZIER CURVE

1) Useful property of a Bezier curve is that it passes through the first & last control points. That is, the boundary conditions at the two ends of the curve are P(0)=p0 & P(1)=pn

2) The curve is a straight line if and only if all the control points are collinear.

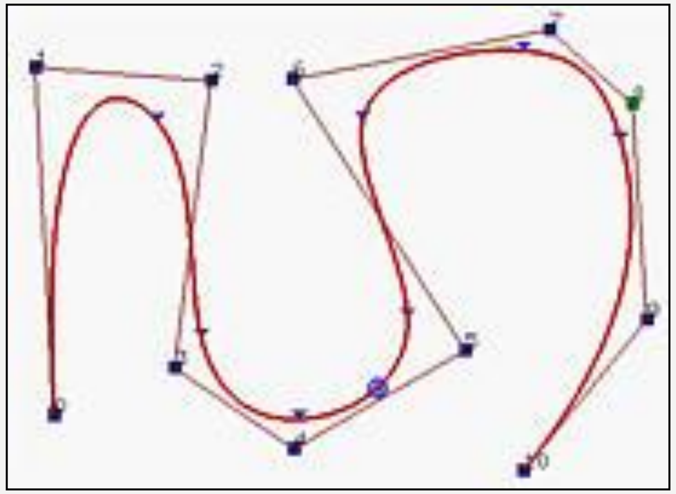
3) The degree of the curves can be determined by the number of control points.(3 control points then degree=3-1=2)

4) It lies within the Convex hull of the control points.

5) Convex hull property ensures that the polynomial smoothly follows the control points without erratic oscillations.

B-SPLINE CURVES

- They are widely used class of approximating splines.
- It has 2 advantages:
  - The degree of B-Spline polynomial can be set independently of the number of control points
  - B-Splines allow local control over the shape of a spline curve or surface.
- In Bezier curve all points are combined to create the curve . This creates lack of control. If only those points nearest to the current parameter are connected can give control. This curve is called as B- spline curve.
- It consists of curve segments whose
- polynomial coefficient only
- depends on few control points.



- we cannot easily control the Bezier curve locally. That is, any change to an individual control point will cause changes in the curve along its full length. In addition, we cannot create a local cusp in the curve, that is, we cannot create a sharp corner unless we create it at the beginning or end of a curve where it joins another curve. Finally, it is not possible to keep the degree of the Bezier curve fixed while adding additional points; any additional points will automatically increase the degree of the curve.
- The so-called B-Spline curve addresses each of these problems with the Bezier curve. It provides the most powerful and useful approach to curve design available today. However, the down side is that the b-spline curve is somewhat more complicated to compute compared to the Bezier curve, but as we shall see the two kinds of curves are closely related. In fact a b-spline curve degenerates into a Bezier curve when the order of the b-spline curve is exactly equal to the number of control points

BEZIER SURFACE

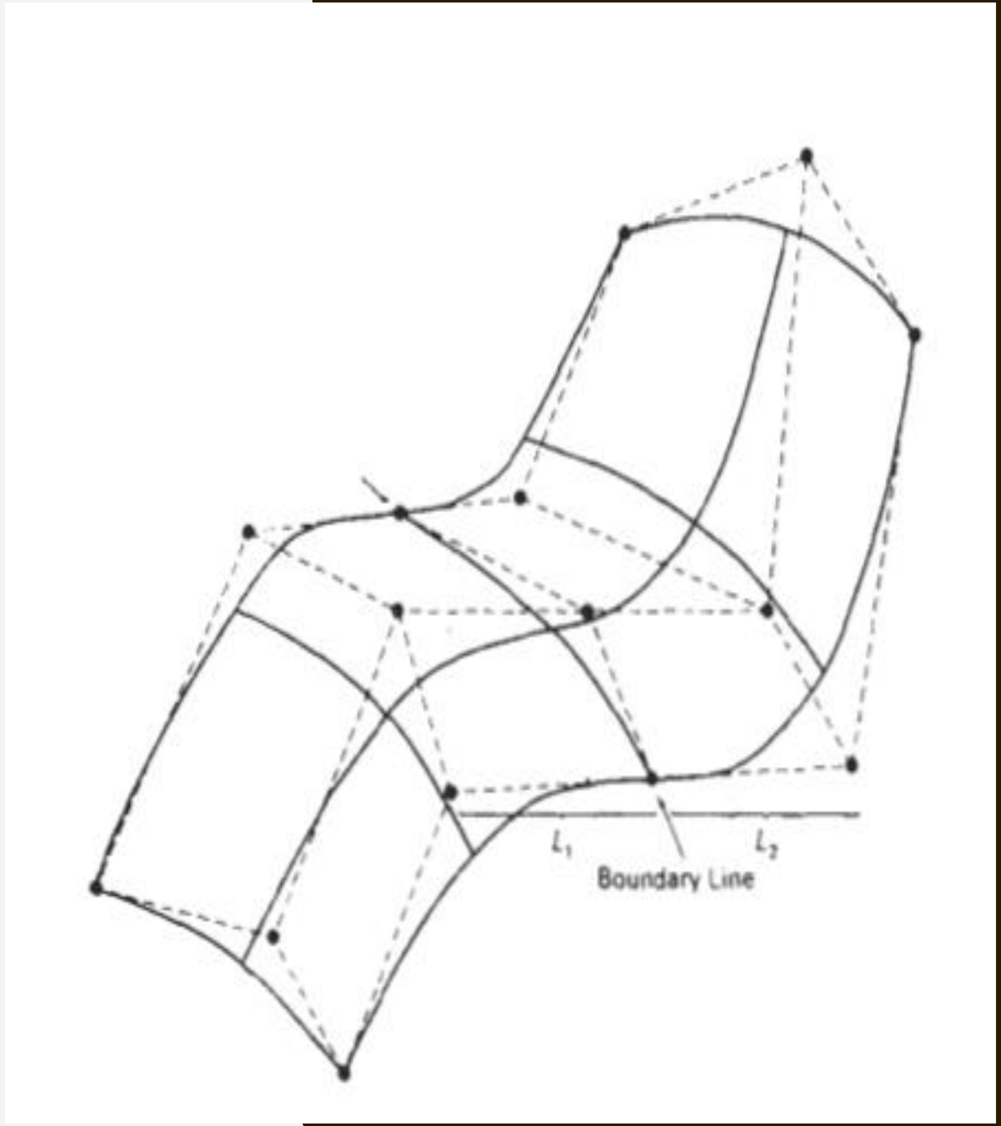
- Bezier surface are an extension of the idea of Bezier curves and share many of their properties.
- Bezier surface is defined by a set of control points.
- Properties are-

1) the surface does not in general pass through the control points (except for the corner points)

2) the surface is contained within the convex hull of the control points.

- Two sets of orthogonal Bezier Curves can be used to design an object surface by specifying an input mesh of control.
- As with the curves, a smooth transition from one section to the other is assured by establishing both zero-order & first-order continuity at the boundary line.

- Zero-order continuity is obtained by matching control points at the boundary.
- First-order continuity is obtained bv choosing control points along a straight line across the boundary and by maintaining a constant ratio of collinear line segments for each set of specified control points across section boundaries.



QUADRIC SURFACE

- A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics)
- Sphere:
- In Cartesian Coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the set of points (x,y,z) that satisfy the equation,

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

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

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

- Torus: A torus is a doughnut-shaped object, as shown in Fig. 10-11. It can be generated by rotating a circle or other conic about a specified axis. The Cartesian representation for points over the surface of a torus can be written in the form,

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

Thank You