Unit 4

Visible-Surface Determination

&

Plane Curves and Surfaces

VISIBLE-SURFACE DETERMINATION

- One of the most **challenging** problems in computer graphics is the **removal of hidden parts** from images of solid objects.
- In real life, the opaque material of these objects obstructs the light rays from hidden parts and prevents us from seeing them.
- In the computer generation, no such automatic elimination takes place when objects are projected onto the screen coordinate system.
- Instead, all parts of every object, including many parts that should be invisible are displayed.
- To remove these parts to create a more realistic image, we must apply a hidden line or hidden surface algorithm to set of objects.
- The algorithm operates on different kinds of scene models, generate various forms of output or cater to images of different complexities.
- All use some form of **geometric sorting** to distinguish visible parts of objects from those that are hidden.

- Geometric sorting locates objects that lie near the observer and are therefore visible.
- Hidden line and Hidden surface algorithms capitalize on various forms of coherence to reduce the computing required to generate an image.
- Different types of coherence are related to different forms of order or regularity in the image.
- Scan line coherence arises because the display of a scan line in a raster image is usually very similar to the display of the preceding scan line.
- A hidden surface algorithm is generally designed to exploit one or more of these coherence properties **to increase efficiency**.
- Hidden surface algorithm bears a strong resemblance to two-dimensional scan conversions.
- When we view a picture containing non-transparent objects and surfaces, then we cannot see those objects from view which are behind from objects closer to eye. We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called **Hidden-surface problem**.

- Types of hidden surface detection algorithms
  - Object space methods
  - Image space methods
- **Object space methods:** In this method, various parts of objects are compared. After comparison visible, invisible or hardly visible surface is determined.
- These methods generally decide visible lines. In the wireframe model, these are used to determine a visible line.
- So these algorithms are line based instead of surface based. Method proceeds by determination of parts of an object whose view is obstructed by other object and draws these parts in the same color.
- **Image space methods:** Here positions of various pixels are determined.
- It is used to locate the visible surface instead of a visible line.
- Each point is detected for its visibility. If a point is visible, then the pixel is on, otherwise off. So the object close to the viewer that is pierced by a projector through a pixel is determined. That pixel is drawn is appropriate color.
- These methods are also called a **Visible Surface Determination**. The implementation of these methods on a computer requires a lot of processing time and processing power of the computer.

Object Space method	Image Space method
1. It deals with object definition directly	1. It is a pixel-based method. It is concerned with the final image, what is visible within each raster pixel.
2. Here surface visibility is determined.	2. Here line visibility or point visibility is determined.
3. It is performed at the precision with which each object is defined, No resolution is considered.	3. It is performed using the resolution of the display device.
4. Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4. Calculations are resolution base, so the change is difficult to adjust.
5. These were developed for vector graphics system.	5. These are developed for raster devices.
6. Object-based algorithms operate on continuous object data.	6. These operate on object data.
7. Vector display used for object method has large address space.	7. Raster systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. There are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.
10. If the number of objects in the scene increases computation time also increases.	10. In this method complexity increase with the complexity of visible parts.

Coherence

- The coherence is defined as the degree to which parts of an environment, or its projection exhibit local similarities. Such as similarities in depth, colour, texture and so on.
- To make algorithms more efficient we can exploit these similarities when we reuse calculations made for one part of the environment or a picture for other nearby parts, either without changes or with some incremental changes. Different kinds of coherence we can use in visible surface algorithms are.

  i. Object coherence: if one object entirely separates from another, comparisons may need to be done only between the two objects, and not between their components faces or edges.

  ii. Face coherence: Usually surface properties vary smoothly across a face. This allows the computations for one part of face to be used with incremental changes to the other parts of the face.

  iii. Edge coherence: The visibility of edge may change only when it crosses a visible edge or penetrates a visible face.

  iv. Implied edge coherence: If one planar face penetrates another their line of intersection can be determined from two points of intersection.

v. Area(window) coherence: A group of adjacent pixels is often belonging to the same visible face.

vi. Span coherence: It refers to a visibility of face over a span of adjacent pixels on a scan line. It is special case area coherence.

viii. Depth coherence: Adjacent parts of the same surface are typically same or very close depth. Therefore, once the depth at one point of the surface is determined, the depth of the points on the rest of the surface can be determined by at the simplest incremental calculation(changes smoothly and slighty).

ix. Frame coherence: Pictures of the same scene at two successive points in time are likely to be quite similar, except small changes in object and view ports. Therefore, the calculations made for one picture can be reused for the next picture in as sequence.

Algorithms used for hidden line surface detection

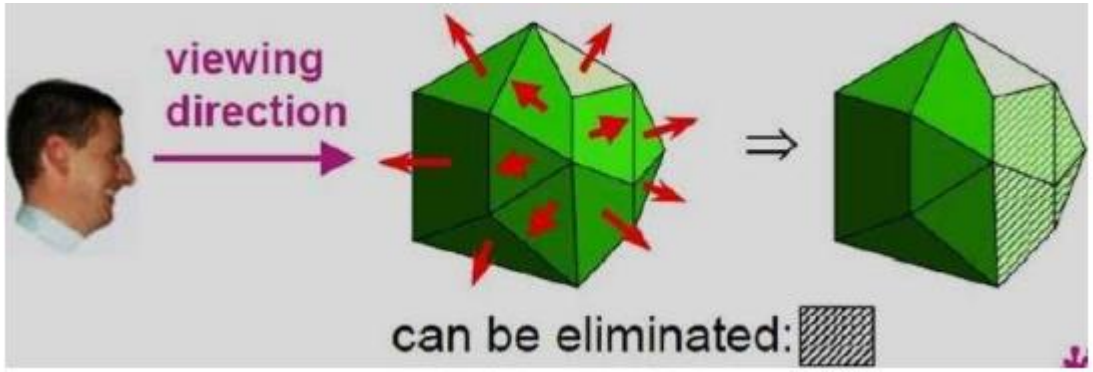
- Back Face Removal Algorithm
- Z-Buffer Algorithm
- Scan Line Algorithm
- Painter Algorithm
- Area Subdivision Algorithm
- Binary Space Partition Tree
- Visible Surface Ray Tracing

Back Face Removal Algorithm

- In a solid object, there are surfaces which are facing the viewer (front faces) and there are surfaces which are opposite to the viewer (back faces).
- These back faces contribute to approximately half of the total number of surfaces. Since we cannot see these surfaces anyway, to save processing time, we can remove them before the clipping process with a simple test.
- Each surface has a normal vector. If this vector is pointing in the direction of the center of projection, it is a front face and can be seen by the viewer.
- If it is pointing away from the center of projection, it is a back face and cannot be seen by the viewer.
- The test is very simple, suppose the z axis is pointing towards the viewer, if the z component of the normal vector is negative, then, it is a back face. If the z component of the vector is positive, it is a front face.

- The direction of the light face can be identified by examining the result N.V > 0
  - Where N: Normal vector to the polygon surface with cartesian components(A,B,C).
  - 1V: A vector in the viewing direction from the eye position.
- We know that, the dot product of two vector gives the product of the length of the two vectors times the cosine of the angle between them.
- This cosine factor is important to us because if the vectors are in the same direction ($0 \leq \theta < \Pi/2$), then the cosine is positive, and the overall dot product is positive. However, if the direction is opposite ($\Pi/2 < \theta <= \Pi$) then the cosine and the overall dot product is negative.
- If the dot product is positive, we can say that the polygon faces towards the viewer; otherwise it faces away and should be removed.

- Note that this technique only caters well for non-overlapping convex polyhedra.
- For other cases where there are concave polyhedra or overlapping objects, we still need to apply other methods to further determine where the obscured faces are partially or completely hidden by other objects (eg., using Depth-Buffer Method or Depth-Sort Method).

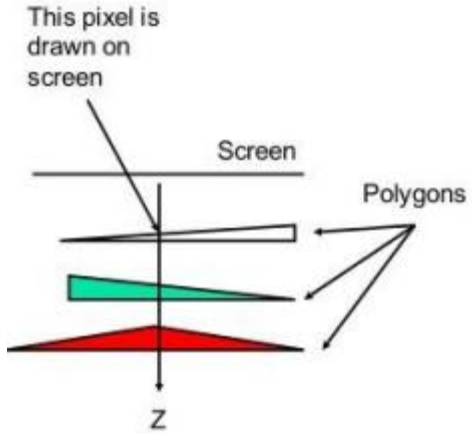


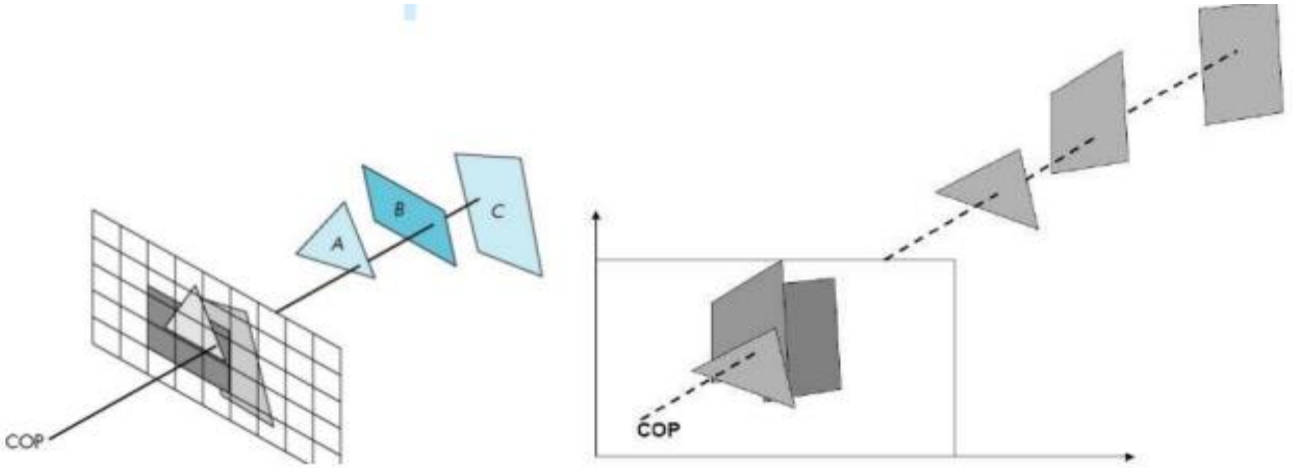
Z-Buffer Algorithm

- One of the simplest and commonly use **image space approach** to eliminate hidden surface is the Z-buffer or depth buffer algorithm. This algorithm compares surface depths at each pixel position on the projection plane.
- The surface depth is measured from the view plane along the z axis of a viewing system.
- When object description is converted to projection coordinates (x, y, z), each pixel position on the view plane is specified by x and y coordinate, and z values gives the depth information.
- Thus, **object depths can be compared by comparing the z- values**.

- The Z-buffer algorithm is usually implemented in the normalized coordinates, so that z values range from 0 at the back-clipping plane to 1 at the front clipping plane.
- The implementation requires another buffer 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.

- The **depth** of each pixel relative to the screen is calculated and saved in a buffer.
- The pixel with the smallest depth is the one that is displayed.
- The other pixel are on surfaces that are hidden.





Algorithm:

1. Initially each pixel of the z-buffer is set to the maximum depth value (the depth of the back clipping plane).
2. The image buffer is set to the background color.
3. Surfaces are rendered one at a time.
4. For the first surface, the depth value of each pixel is calculated.
5. If this depth value is smaller than the corresponding depth value in the z-buffer (ie. it is closer to the view point), both the depth value in the z-buffer and the color value in the image buffer are replaced by the depth value and the color value of this surface calculated at the pixel position.
6. Repeat step 4 and 5 for the remaining surfaces.
7. After all the surfaces have been processed, each pixel of the image buffer represents the color of a visible surface at that pixel.

- This method requires an additional buffer (if compared with the Depth-Sort Method) and the overheads involved in updating the buffer. So this method is less attractive in the cases where only a few objects in the scene are to be rendered.
- Simple and does not require additional data structures.
- The z-value of a polygon can be calculated incrementally.
- No pre-sorting of polygons is needed.
- No object-object comparison is required.
- Can be applied to non-polygonal objects.
- Hardware implementations of the algorithm are available in some graphics workstation.
- For large images, the algorithm could be applied to, eg., the 4 quadrants of the image separately, so as to reduce the requirement of a large additional buffer.

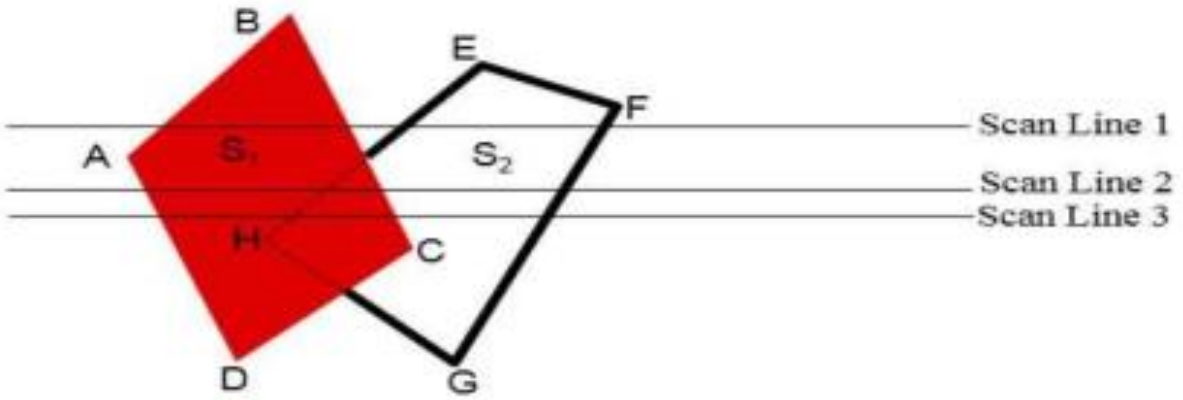
- Advantages:
  1. It is easy to implement
  2. It can be implemented in hardware to overcome the speed problem.
  3. Since the algorithm processes objects one at a time, the total number of polygons in a picture can be arbitrarily large.
- Disadvantages:
  1. It requires an additional buffer and hence the large memory.
  2. It is time consuming process as it requires comparison for each pixel instead of for the entire polygon.

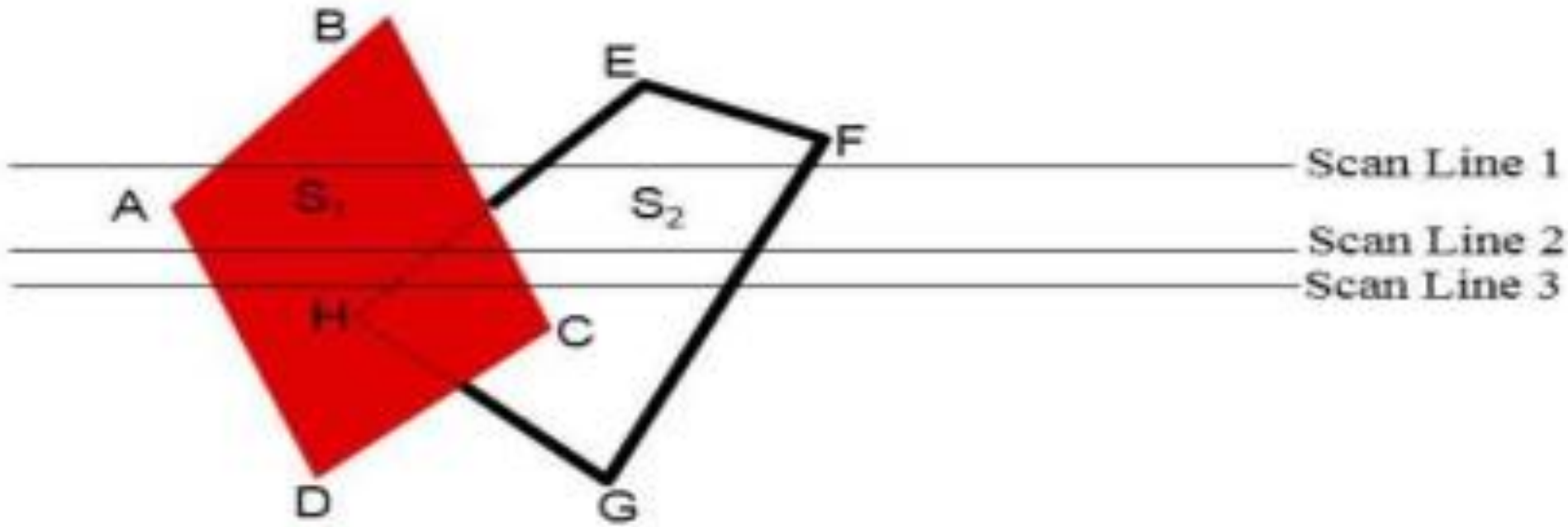
Scan Line Algorithm

- A scan line method of hidden surface removal is another approach of image space method.
- It is an extension of the scan line algorithm for filling polygon interiors.
- Here, the algorithm deals with more than one surfaces. As each scan line is processed, it examines all polygon surfaces intersecting that line to determine which are visible. It does the **depth calculation** and finds which polygon is nearest to the view plane. Finally, it enters the intensity value of the nearest value of the nearest polygon at that position into the frame buffer.

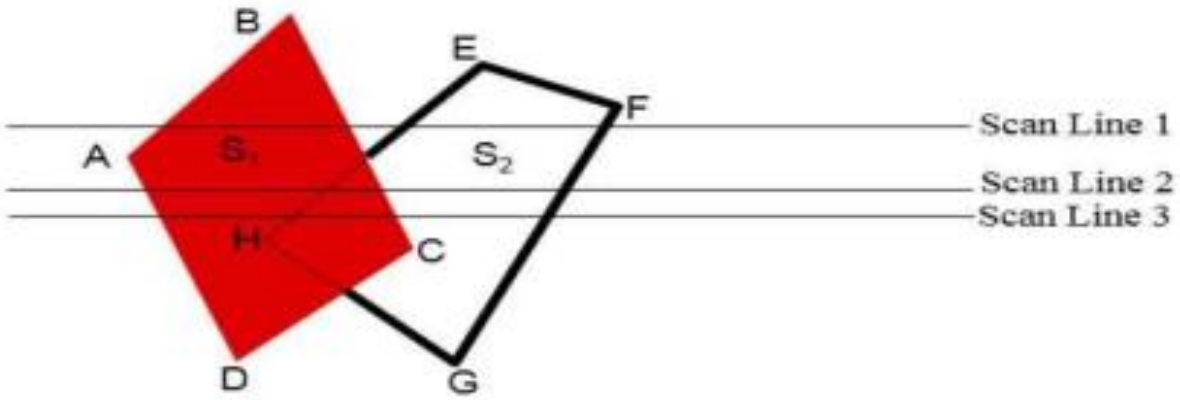
- We know that scan line algorithm maintains the **active edge list**. This active edge list contains only edges that cross the current scan line, sorted in order of increasing order.
- The scan to indicate whether a position along a scan line is inside or outside of the surface. Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned ON, and at the rightmost boundary, it is turned OFF.

- Example 1: The figure illustrates the scan line method for hidden surface removal.
- The active edge list for scan line 1 contains the information for edges AD, BC, EH an FG.
- For the positions along this scan line between edges AD and BC, only the flag for surface S1 is ON. Therefore, no depth calculations are necessary, and intensity information for surface S1 is entered the frame buffer.
- Similarly, between edges EH and FG, only the flag for surface S2 is ON and during that portion of scan line intensity information for surface S2 is entered the frame buffer.




B
E
F
A
S1
S2
H
C
D
G
Scan Line 1
Scan Line 2
Scan Line 3

- For scan line, the active edge list contains edges AD, EH, BC, FG. Along the scan line 2 from edge AD to edge EH, only the flag for surface S1 is ON.
- However, between edges EH and BC, the flags for both surface are ON. In this portion of scan, line 2, the depth calculations are necessary.
- Here we have assumed that the depth of S1 is less than the depth of S2 and hence the intensities of surface S1 are loaded into the frame buffer.
- Then for edge BC to edge FG portion of scan line 2 intensities of surface S2 are entered the frame buffer because buffer during that portion only flag for S2 is ON.



- This algorithm is based on the Image-space method and concept of coherence. As its name suggests itself Scan-line algorithm, so it processes one line at a time rather than processing one pixel(a point on raster display) at a time.
- Edge list table: This list maintains the record of all the edges by storing their endpoint coordinates. The x-coordinate that we choose, whose Y-coordinate = Ymin.
- 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(list): This list consists of:
  a. Polygon Id.
  b. Plane equation.
  c. Color Information of the surface.
  d. Flag of surface(on/off)].

Painter’s algorithm (Depth sort Algorithm)

- The techniques used by **Painter’s algorithm** are **image space** and **object space**.
- The name of this algorithm is Painter’s because it’s working is like a painter who creating an **oil painting**. Just like an artist paints, he start his painting with an empty canvas, the first thing the artist will do is to create a **background layer** for the painting, after this layer he start creating **another layers** of objects **one-by-one**. In this way he completes his painting, by covering the previous layer partially or fully according to the requirement of the painting.
- This algorithm is basically used to paint the **polygons** in the view plane by considering their distance from the viewer. The polygons which are at more distance from the viewer are painted first. After that the nearer polygons are started painted on or over more distant polygons according to the requirement.

- In this algorithm the polygons or surfaces in the scene are firstly scanned or then painted in the frame buffer in the **decreasing distance** from view point of the viewer starting with the polygons of **maximum depth**.
- Firstly the **depth sort** is performed in which the polygons are listed according to their **visibility order** or depth priority.
- As this algorithm uses the concept of depth priority so it is also called as **depth priority algorithm** or **priority algorithm**.
- The frame buffer is painted with the background color. After that the polygon which is farthest enter to the frame buffer. For this, the pixel information will get changed i.e. information of the background which has the farthest polygon get replaced with that of the background. This is going to be repeatedly changed as we move from one polygon to the other and end up with the nearest polygon.

- To find out which rectangle is to be overlap, we need to find the minimum and maximum x and y values to the rectangles which are going to test for overlapping. If the minimum value of y of one of the rectangles is larger than the maximum y value of the other rectangle then they are not going to be overlapped and as the rectangles are not overlapped then the surfaces are also not overlapped.
- The same test is to be performed for the x-coordinates.
- If the surfaces are overlapping, we do not know which surface should be present on the top of the other. To find out which surface is to be present on the top the **principle of mini-max** is used on the depth values of both the overlapping surfaces.

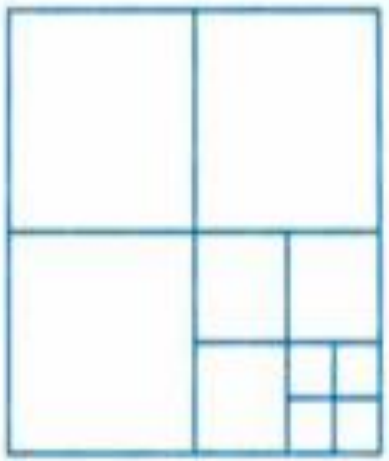
Distant Background is painted first

Objects closer than the background are painted

Finally, Nearest Objects are painted

Warnock's Algorithm (Area Subdivision Algorithm)

- An interesting approach to the hidden-surface problem was developed by Warnock.
- He Developed area subdivision algorithm which subdivides each area into four equal squares.
- At each stage in the recursive-subdivision process, the relationship between projection of each polygon and the area of interest is checked for four possible relationships:
  1. Surrounding polygon – One that completely encloses the (shaded) area of interest.
  2. Overlapping or Intersecting Polygon –One that is partly inside and partly outside the area.
  3. Inside or Contained Polygon – One that is completely inside the area.
  4. Outside or Disjoint Polygon- One that is completely outside the area.



Dividing a square area into equal-sized quadrants at each step.

Possible relationships between polygon surfaces and a rectangular area.

- After checking four relationship we can handle each relationship as follows:

  1. If all the polygons are disjoint from the area, then the background colour is displayed in the area.

  2. If there is only one intersecting or only one contained polygon, then the area is first filled with background colour, and then the part of the polygon contained in the area is filled with colour of polygon.

  3. If there is a single surrounding polygon, but no intersecting or contained polygons, then the area is filled with the colour of the surrounding polygon.

  4. If there are more than one polygon intersecting, contained in, or surrounding the area then we must do some more processing.

a) The four intersections of surrounding polygon are all closer to the viewpoint than any of the other intersections. Therefore, the entire area is filled with the colour of the surrounding polygon.

b) That surrounding polygon is not completely in front of the intersecting polygon. The Warnock's algorithm stops subdivision of area only when the problem is simplified or even area is only a single pixel.

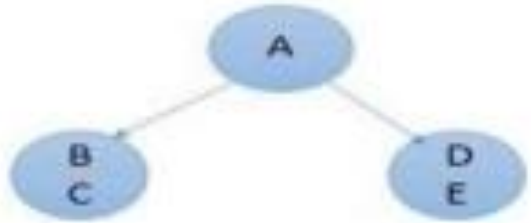
- Algorithm:
  1. Initialize the area to be the whole screen.
  2. Create the list of polygons by sorting them with their z-values of vertices. Don’t include disjoint polygons in the list because they are not visible.
  3. Find the relationship of each polygon.
  4. Perform the visibility decision test
     a. If all the polygons are disjoint from the area, then fill area with background colour.
     b. If there is only one intersecting or only one contained polygon then first fill entire area with background colour and then fill the part of the polygon contained in the area with the colour of polygon.
     c. If there is a single surrounding polygon, but no intersecting or contained polygons, then fill the area with the colour of the surrounding polygon.
     d. If the surrounding polygon is closer to the viewpoint than all other polygons, so that all other polygons are hidden by it, fill the area with the colour of the surrounding polygon.
     e. If the area is the pixel (x, y) and neither a, b, c, nor d apply, compute the z coordinate at pixel (x, y) of all polygons in the list. The pixel is then set to colour of the polygon which is closer to the viewpoint.
  5. If none of the above tests are true, then subdivide the area and go to step 2

- Advantages:
  1. It follows the divide-and-conquer strategy; therefore, parallel computers can be used to speed up the process.
  2. Extra memory buffer is not required.

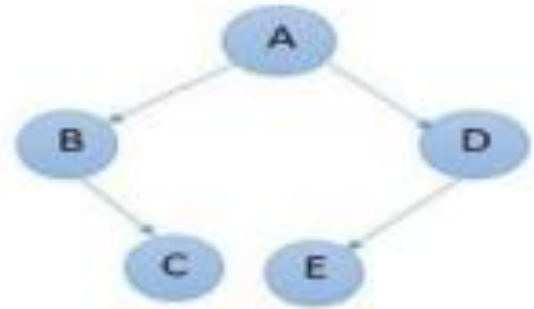
Binary Space Partition (BSP) Trees:

- Binary space partitioning is used to calculate visibility.
- It is suitable for a static group of 3D polygons to be viewed rom a number of view points.
- 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 into 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.

(a)



(b)

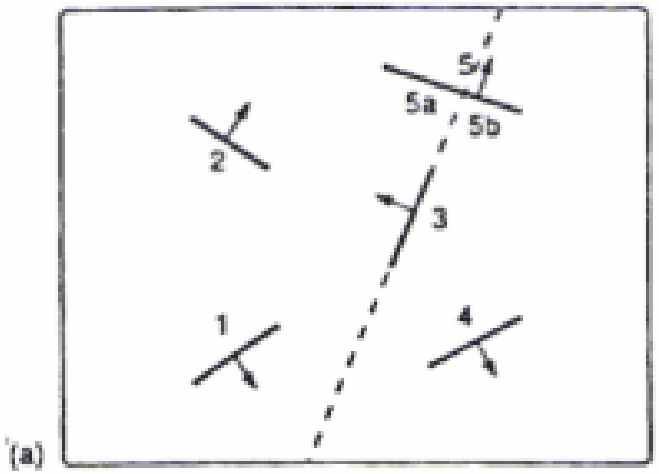


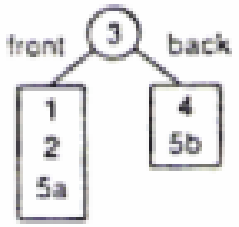
(c)

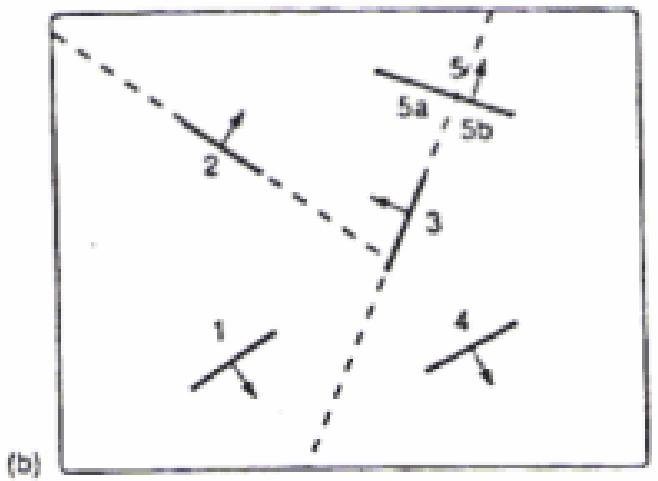
(d)

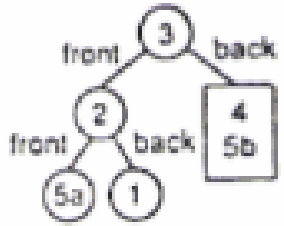
BSP trees

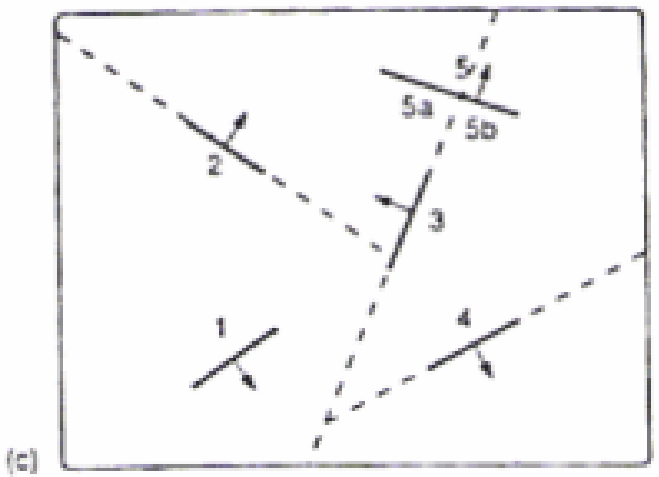
- From the above figure, first take A as a root.
- Make a list of all nodes in figure (a).
- Put all the nodes that are in front of root A to the left side of node A and put all those nodes that are behind the root A to the right side as shown in figure (b).
- Process all the front nodes first and then 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.

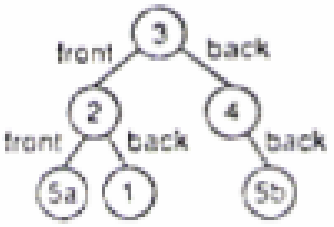

(a)
5
5a
5b
2
3
1
4


front
3
back
1
2
5a
4
5b


(b)
5
5a
5b
2
3
1
4


front
3
back
front
2
back
4
5b
5a
1


(c)
5
5a
5b
2
3
1
4

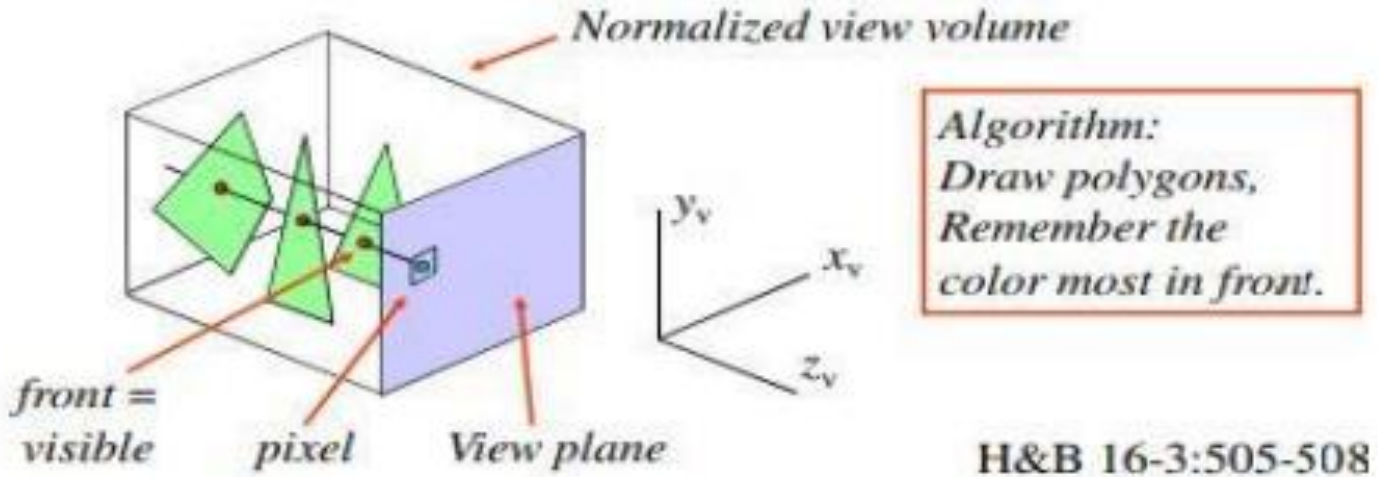

front
3
back
front
2
back
4
back
5a
1
5b

1. The algorithm first builds the BSP tree:
   - a root polygon is chosen (arbitrarily) which divides the region into 2 half-spaces (2 nodes => front and back)
   - a polygon in the front half-space is chosen which divides the half-space into another 2 half-spaces
   - the subdivision is repeated until the half-space contains a single polygon (leaf node of the tree)
   - - the same is done for the back space of the polygon.
2. To display a BSP tree:
   - see whether the viewer is in the front or the back half-space of the root polygon.
   - if front half-space then first display back child (subtree) then itself, followed by its front child/ subtree
   - the algorithm is applied recursively to the BSP tree.

- Back face removal is achieved by not displaying a polygon if the viewer is located in its back half-space.
- It is an object space algorithm (sorting and intersection calculations are done in object space precision)
- If the view point changes, the BSP needs only minor re-arrangement.
- A new BSP tree is built if the scene changes
- The algorithm displays polygon back to front (cf. Depth-sort)

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.

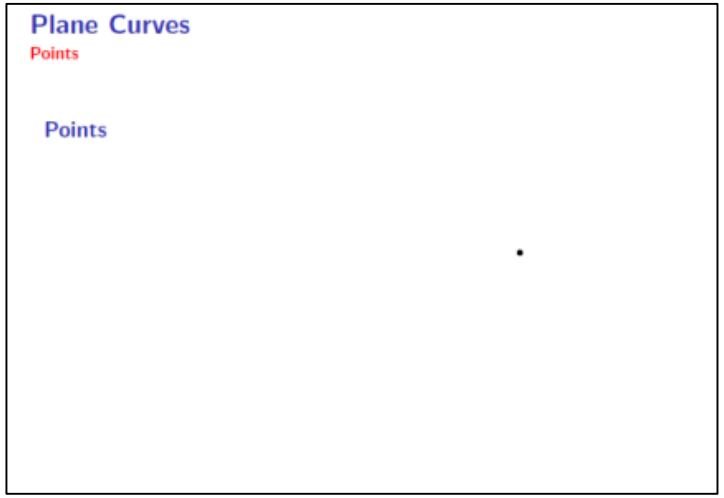


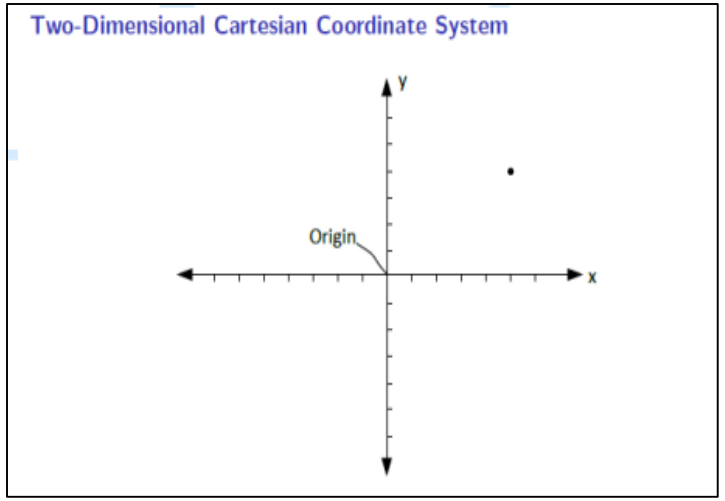
Visible-Surface Ray Tracing

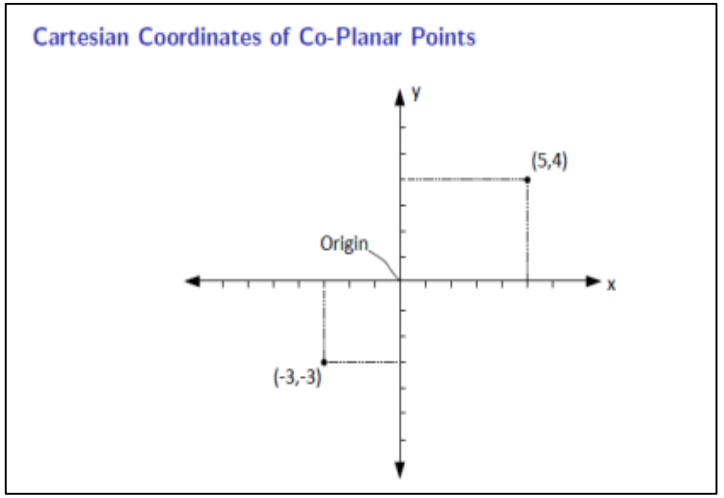
INTRODUCTION TO CURVE

- The world around us is full of objects of remarkable shapes.
- Nevertheless, in computer graphics, we continue to populate our virtual worlds with flat objects.
- Graphics system scan render flat three –dimensional polygons at high rates, including doing hidden-surface removal, shading, and texture mapping.
- We introduce three ways to model curves and surfaces, paying most attention to the parametric polynomial forms.

Curve and Surface Representation Methods

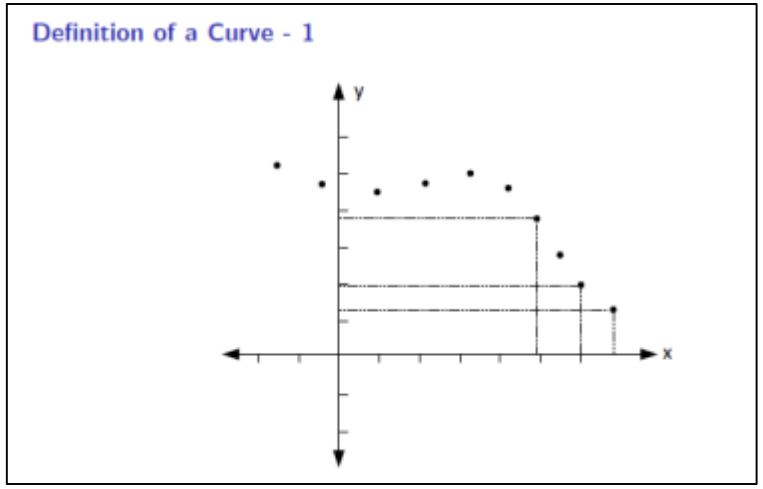


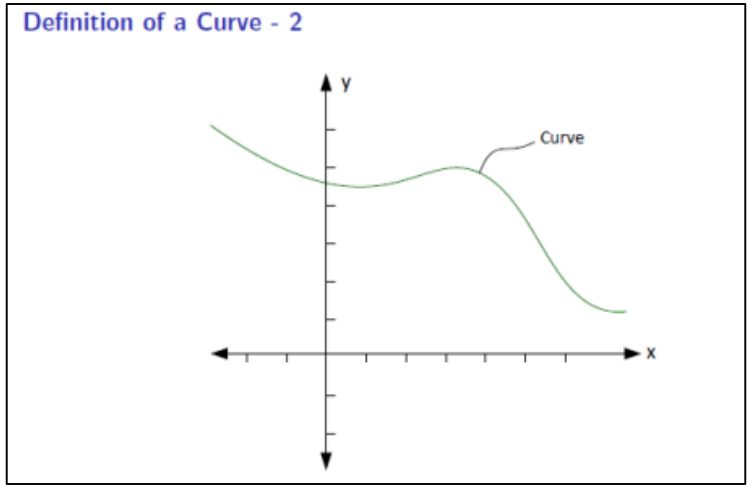


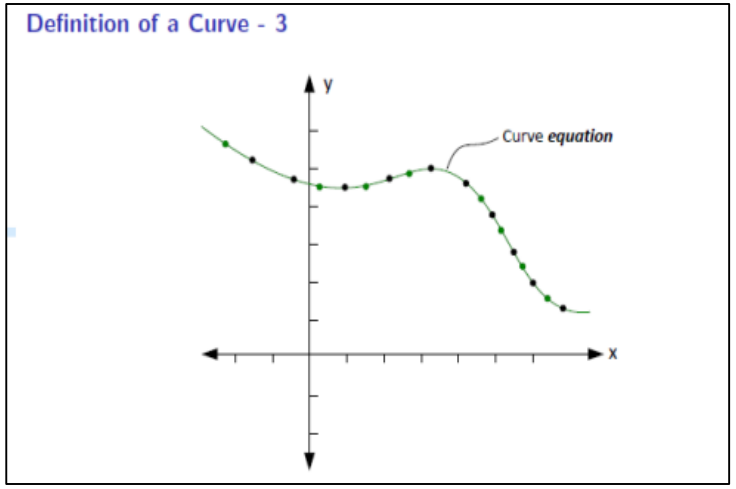


- A point is a 0 – dimensional mathematical object. A point is the building element of higher – dimensional geometric objects (line, plane , etc.)
- A coordinate system is a system which is used to define the location of a point uniquely with respect to a point of origin.

- A curve is the path of a point in motion.







Implicit curve

- an implicit curve is defined by an implicit function.
- Implicit function is a function defined for differentiation of functions containing the variables, which cannot be easily expressed in the form of y = f(x).
- The function of the form **g(x, y)=0** or an equation, $x^2 + y^2 + 4xy + 25 = 0$ is an example of **implicit function**, where the dependent variable 'y' and the independent variable 'x' cannot be easily segregated to represent it as a function of the form y = f(x).
- Implicit function is a function with multiple variables, and one of the variables is a function of the other set of variables.
- A function **f(x, y) = 0** such that it is a function of x, y, expressed as an equation with the variables on one side, and equalized to zero.

- In 2D, an implicit curve can be represented by the equation f(x,y) = 0
- In 3D, an implicit curve can be represented by the equation f(x,y,z) = 0
- It can represent multivalued curves (multiple y values for an x value). A common example is the circle, whose implicit representation is x2 + y2 - R2 = 0

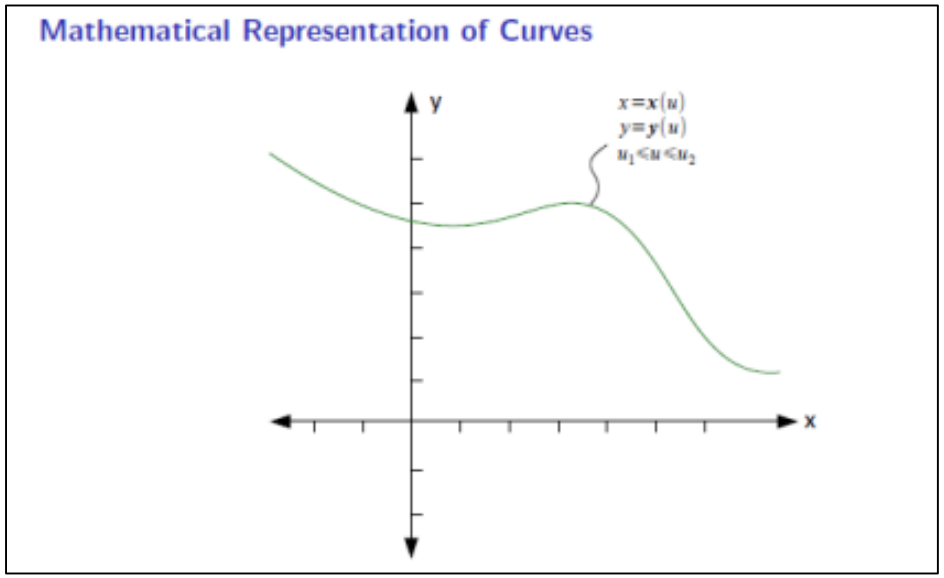
Explicit Curves

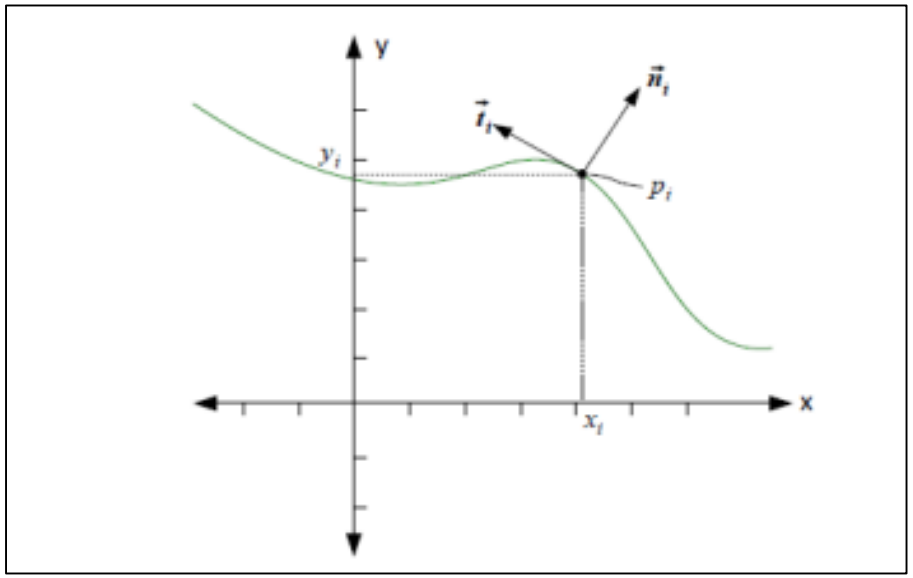
- A mathematical function **y = f(x)** can be plotted as a curve.
- Such a function is the explicit representation of the curve.
- The explicit representation is not general, since it cannot represent vertical lines and is also single-valued.
- For each value of x, only a single value of y is normally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

Parametric Curves

- Curves having parametric form are called parametric curves.
- The explicit and implicit curve representations can be used only when the function is known.
- In practice the parametric curves are used.
- A two-dimensional parametric curve has the following form −
- P(t) = f(t), g(t) or P(t) = x(t), y(t)
- The functions f and g become the (x, y) coordinates of any point on the curve, and the points are obtained when the parameter t is varied over a certain interval [a, b], normally [0, 1].

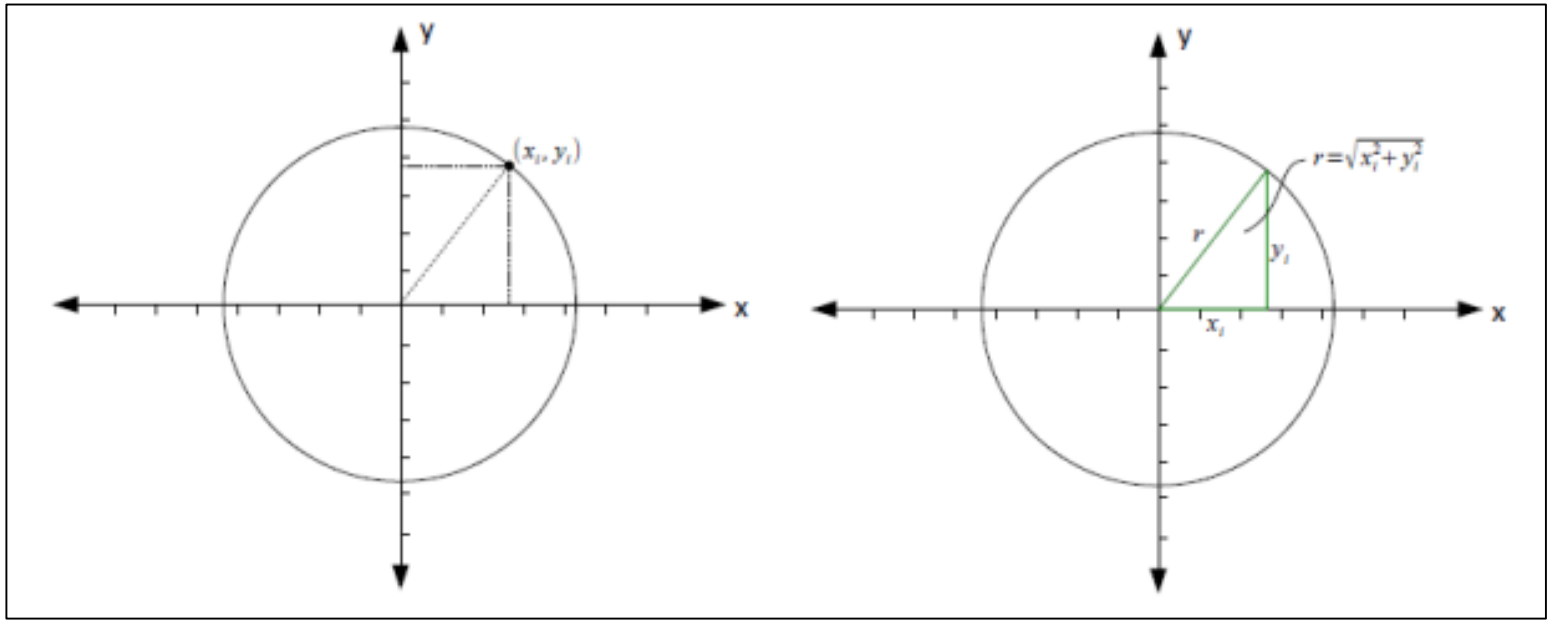
- Tangent and normal vectors of planar curve at point pi.





Circle

- A circle is the path of a point that moves such that it is at a constant distance from a given co-planar point.



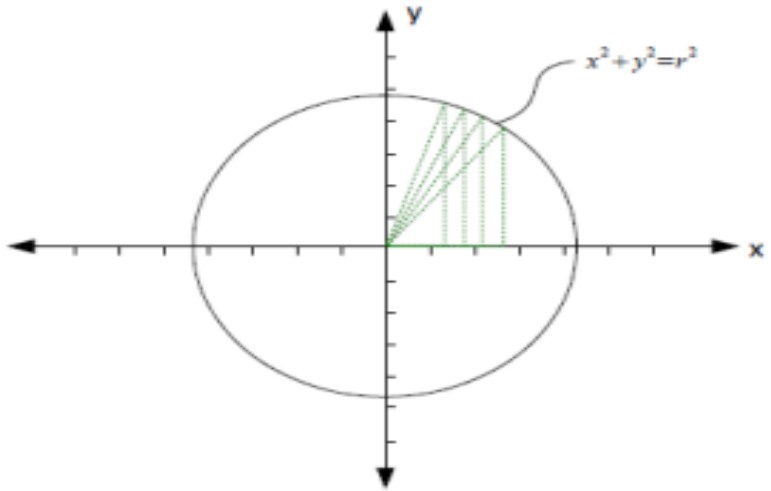
- Let us consider the circle's center as point O, and line OP is the radius equal to r. It has its center at the origin (0, 0). In Cartesian coordinates, the equation of a circle with a point (x, y) on it is represented as follows:

$$\mathbf{x^2 + y^2 = r^2}$$

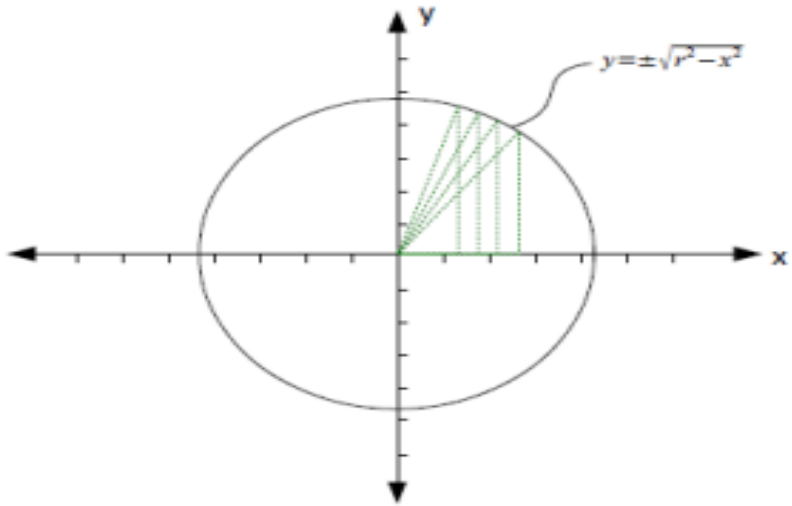
- Let us consider the point P (x, y) as the coordinates of any point on the circle. We get a right-angled triangle when a perpendicular line is drawn from the point P to the X-axis. Here, an acute angle is formed opposite to the perpendicular line represented as θ. It is called a parameter.
- With the help of trigonometric ratios, we get the following values of x and y:

$$\mathbf{y = r \sin \theta \quad and \quad x = r \cos \theta.}$$

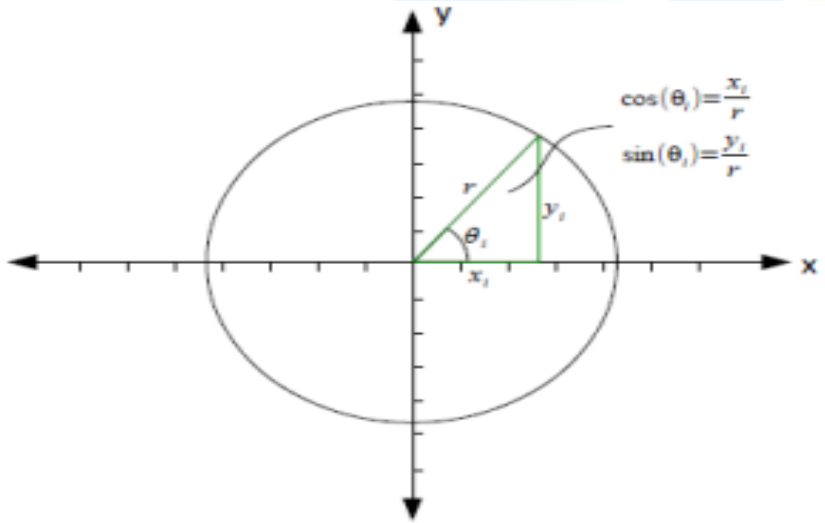
- Therefore, the parametric equation of a circle that is centered at the origin (0,0) can be given as **P (x, y) = P (r cos θ, r sin θ),** (Here $0 \leq \theta \leq 2\pi$.)
- In other words, it can be said that for a circle centered at the origin, $x^2 + y^2 = r^2$ is the equation with $y = r \sin \theta$ and $x = r \cos \theta$ as its solution. Here, θ is the parameter.



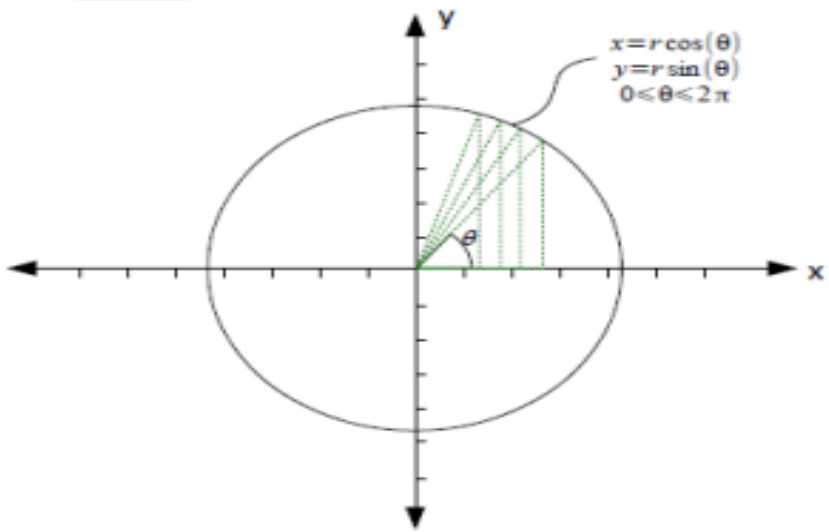
Implicit equation of a circle.



Explicit equation of a circle.



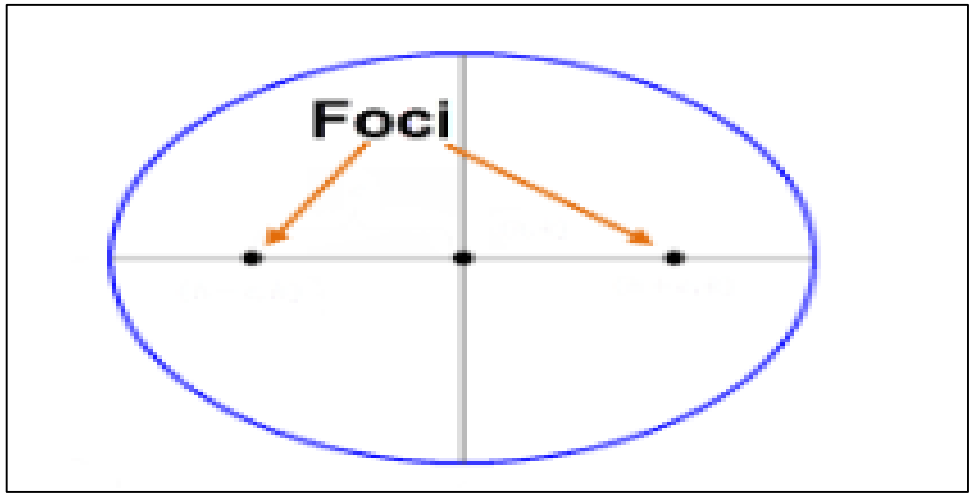
Derivation of parametric representation.



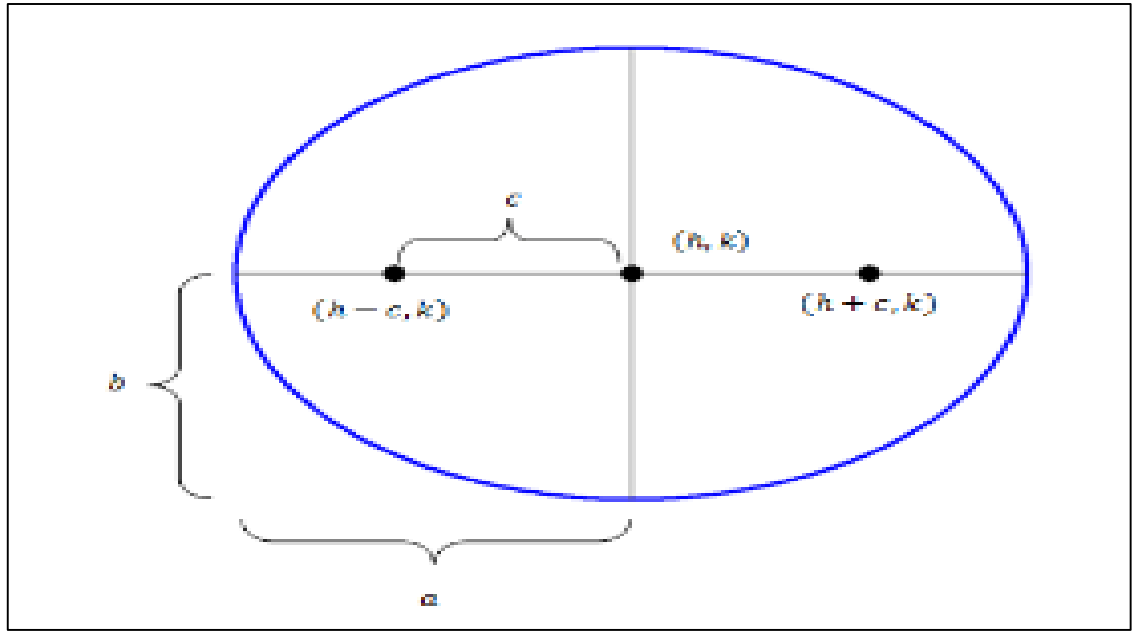
Parametric representation.

PARAMETRIC REPRESENTATION OF ELLIPSE

- An **ellipse** is a rounded shape with two **foci**.
- Every coordinate on the ellipse can be described by its distance from the two foci. While a given point's distance from each focus is unique, the sum of the two distances is the same for every point on the ellipse.



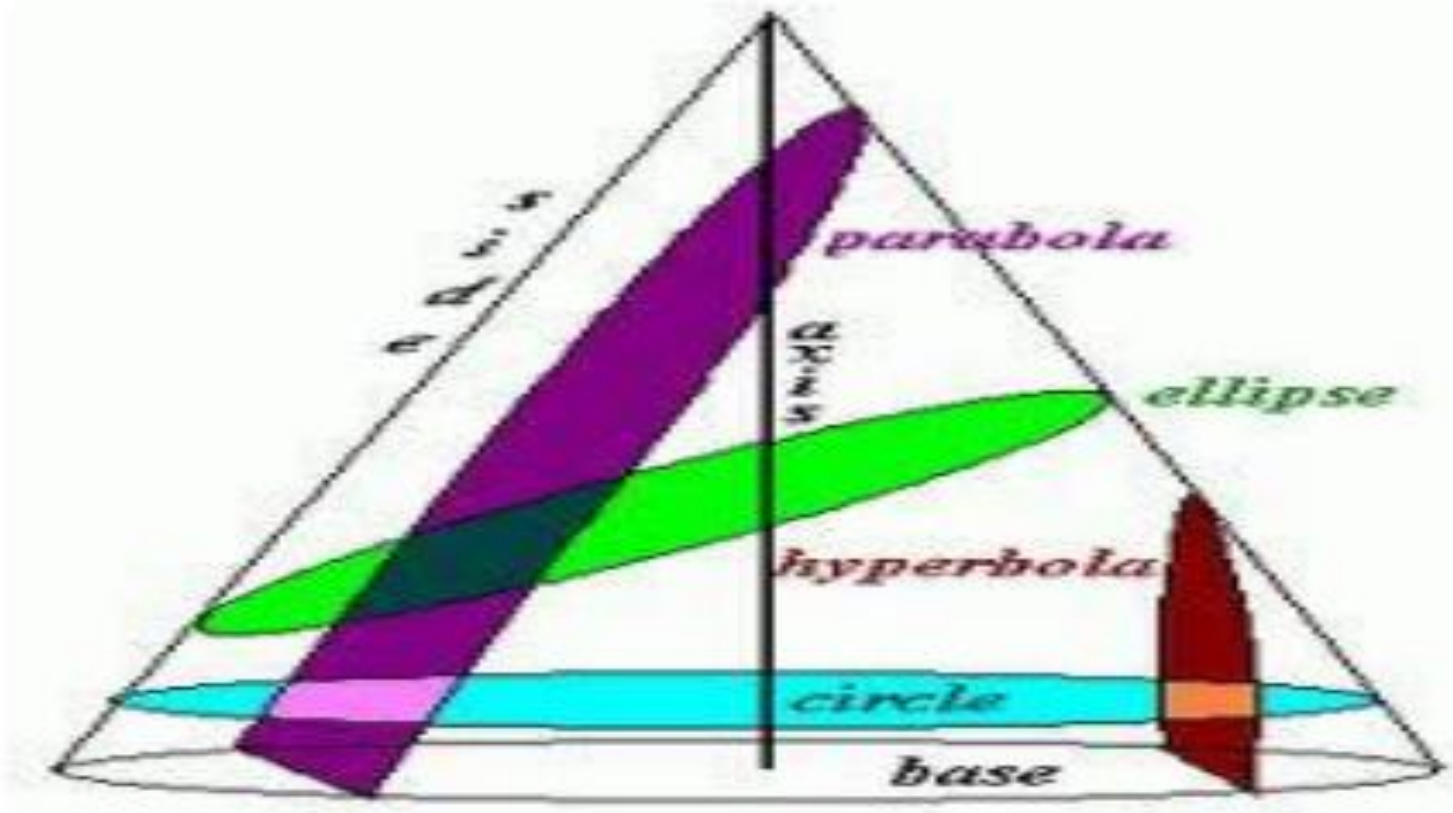
- The standard equation for an ellipse is $\mathbf{(x-h)^2/a^2 + (y-k)^2/b^2 = 1}$, where ($h$,k) is the center of the ellipse, and 2a and 2b are the lengths of the axes of the ellipse. The longer axis is called the **major axis**, while the shorter axis is called the **minor axis.**



- A **circle** is a special type of ellipse where a is equal to b. You write the standard equation for a circle as $(x-h)^2+(y-k)^2=r^2$, where r is the **radius** of the circle and (h,k) is the **center** of the circle.

- The **parametric form for an ellipse** is F(t) =(x(t),y(t)) where **x(t) = acost(t) +h and y(t) = bsin(t) +k.**
- Since a circle is an ellipse where both foci are in the center and both axes are the same length, the **parametric form of a circle** is

F(t) =(x(t),y(t)) where x(t) = rcost(t) +h and y(t) = rsin(t) +k.


parabola
axis
ellipse
hyperbola
circle
base

PARAMETRIC REPRESENTATION OF PARABOLA

- The best form to represent the parametric coordinates of any point on the parabola with the standard equation **$y^2 = 4ax$ is $(at^2, 2at)$.**
- And for all the values of t, the coordinates is $(at^2, 2at)$. All the values of t will satisfy the equation of the parabola that is $y^2 = 4ax$.

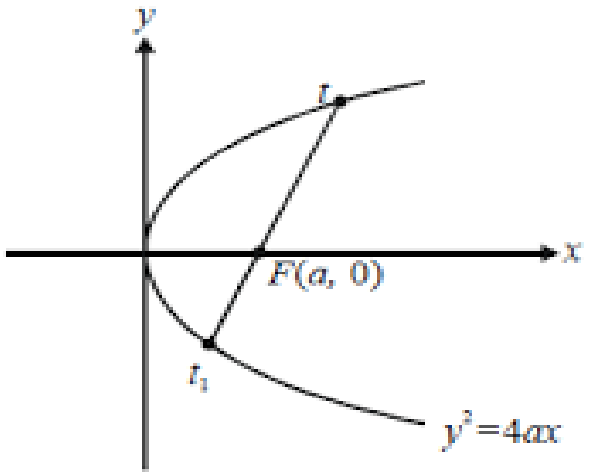
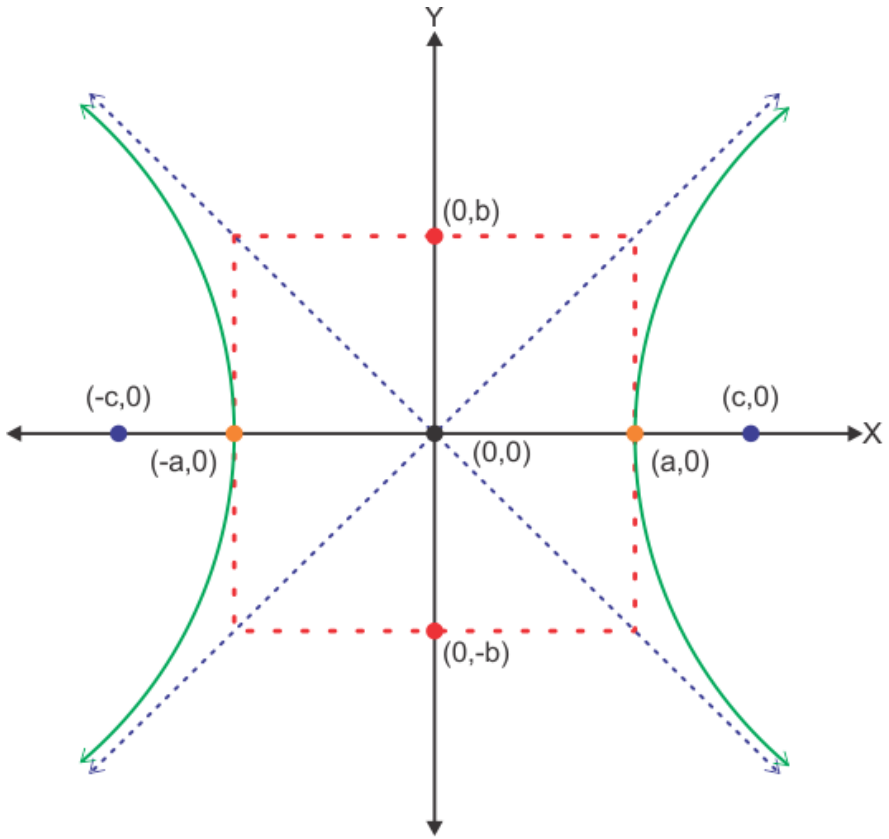


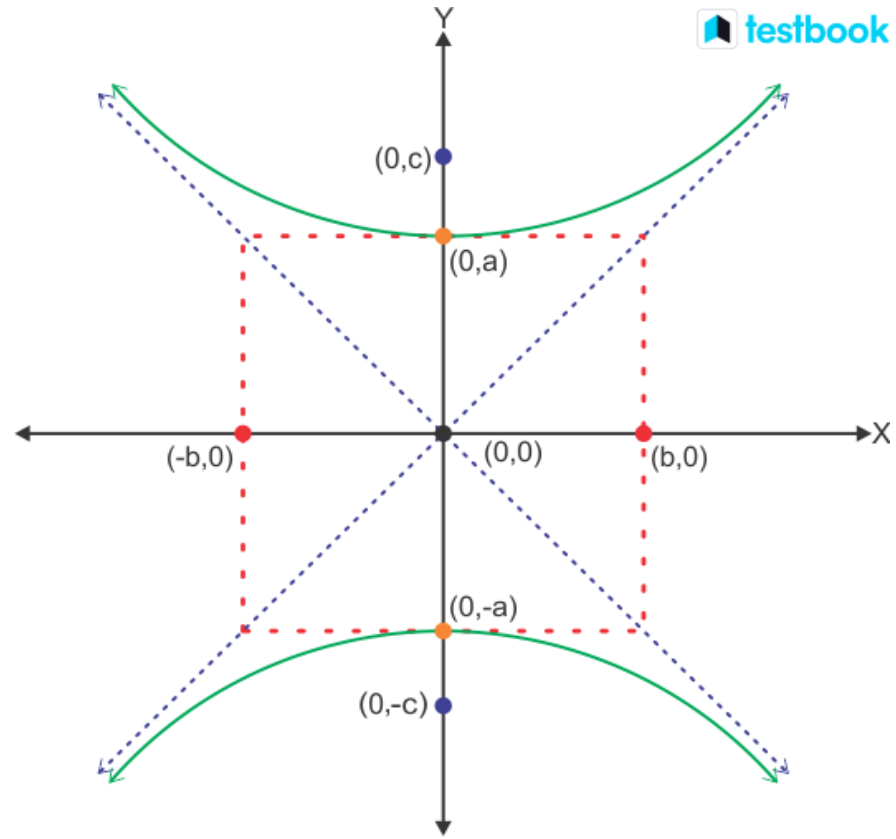
Fig - 17

PARAMETRIC REPRESENTATION OF HYPERBOLA

- hyperbola is a collection of points whose difference in distances from two foci is a fixed value. This difference is obtained from the distance of the farther focus minus the distance of the nearer focus.
- Standard equation of hyperbola with center (0,0) and transverse axis on the x-axis and the conjugate axis is the y-axis is as shown:
- Form: $x^2/a^2 - y^2/b^2 = 1$
- In this form of hyperbola, the center is located at the origin and foci are on the X-axis.



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



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

- Standard equation of a hyperbola with center (0,0) and transverse axis on the y-axis and its conjugate axis is the x-axis is as shown:
- Form: $y^2/a^2 - x^2/b^2 = 1$
- In this form of hyperbola, the center is located at the origin and foci are on the Y-axis.

- Standard equation of a hyperbola with center (h,k) and transverse axis parallel to the x-axis is as shown:
- $(x-h)^2/a^2 - (y-k)^2/b^2 = 1$
- Standard equation of a hyperbola with center (h,k) and transverse axis parallel to the y-axis is as shown:
- $(y-k)^2/a^2 - (x-h)^2/b^2 = 1$

- For the hyperbola $\mathbf{x^2/a^2 - y^2/b^2 = 1.}$
- The parametric equation is $x=a\sec\theta$, $y=b\tan\theta$
- and parametric coordinates of the point resting on it are presented by $\mathbf{(a\sec\theta, b\tan\theta).}$

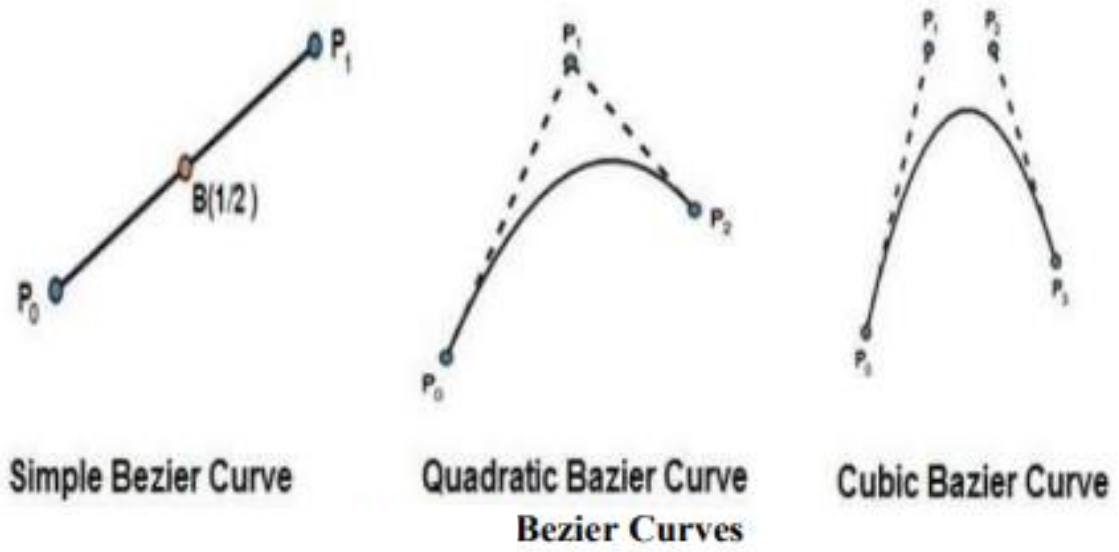
Bezier curve

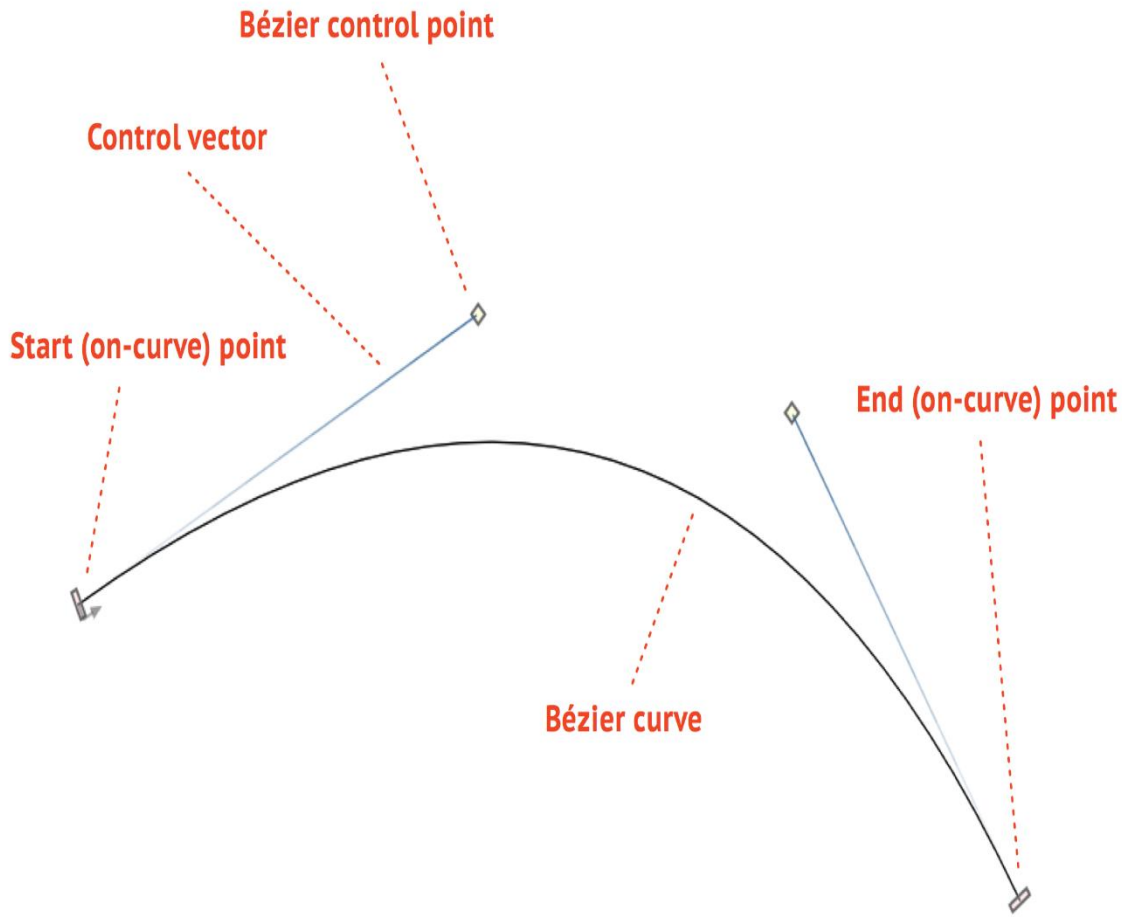
- Bezier curve is discovered by the French engineer Pierre Bézier.
- These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as −
- ${}^{n}\sum_{k=0} P_i B_i^n (t)$ Where p_i is the set of points and $B_i^n (t)$ represents the Bernstein polynomials which are given by −

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

- Where n is the polynomial degree, i is the index, and t is the variable.

- The simplest Bezier curve is the straight line from the point P0 to P1.
- A quadratic Bezier curve is determined by three control points.
- A cubic Bezier curve is determined by four control points.




Bézier control point
Control vector
Start (on-curve) point
End (on-curve) point
Bézier curve

- Bezier curves have the following properties:
  - They generally follow the shape of the control polygon, which consists of the segments joining the control points.
  - They always pass through the first and last control points.
  - They are contained in the convex hull of their defining control points.
  - The degree of the polynomial defining the curve segment is one less that the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
  - A Bezier curve generally follows the shape of the defining polygon.
  - The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
  - The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
  - No straight line intersects a Bezier curve more times than it intersects its control polygon.
  - They are invariant under an affine transformation.
  - Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.

B-Spline Curves

- The Bezier-curve produced by the Bernstein basis function has limited flexibility.
  - First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.
  - The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.
- The B-spline basis contains the Bernstein basis as the special case.

- The B-spline basis is non-global. A B-spline curve is defined as a linear combination of control points Pi and B-spline basis function Ni, k (t) given by $N_{i,\,k}$
- $C(t)=\sum^{n}_{i=0} P,\ N_{i,k}(t),\ n \geq k-1,\ t \epsilon [tk-1, tn+1]$
- Where,
- {pi: i=0, 1, 2….n} are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree k - 1,
- the $N_{i,k}(t)$ are the “normalized B-spline blending functions”. They are described by the order k and by a non-decreasing sequence of real numbers normally called the “knot sequence”.

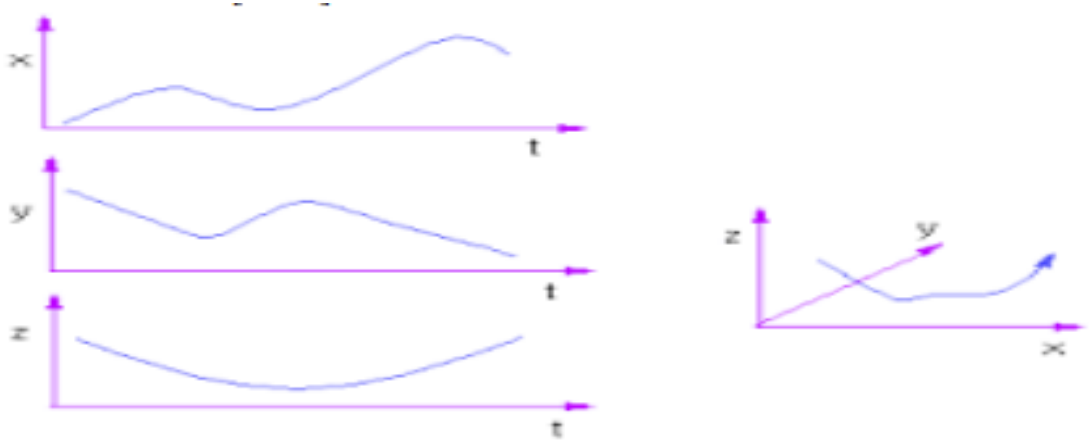
- t_i:i=0,...n+K
- The Ni , k functions are described as follows −
- $N_{i,1}(t)= \begin{cases} 1, \text{ ifu}\epsilon[ti,ti+1) \\ 0, \text{ Otherwise} \end{cases}$

- and if k > 1,
- $N_{i,k}(t)=(t-t_i/t_{i+k-1})\ N_{i,k-1}(t)+ (t_{i+k}-t/t_{i+k}-t_{i+1})\ N_{i+1,k-1}(t)$
- and $t\epsilon[t_{k-1},t_{n+1})$

- Properties of B-spline Curve:
  - B-spline curves have the following properties:
  - The sum of the B-spline basis functions for any parameter value is 1.
  - Each basis function is positive or zero for all parameter values.
  - Each basis function has precisely one maximum value, except for k=1.
  - The maximum order of the curve is equal to the number of vertices of defining polygon.
  - The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
  - B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
  - The curve exhibits the variation diminishing property.
  - The curve generally follows the shape of defining polygon.
  - Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
  - The curve line within the convex hull of its defining polygon

Parametric Cubic curves

- Cubic curves are commonly used in graphics because curves of lower order commonly have too little flexibility, while curves of higher order are usually considered unnecessarily complex and make it easy to introduce undesired wiggles.
- A parametric cubic curve in 3D is defined by: usually , we consider t = [0…1].



Cubic Curves

Quadratic Surfaces

- A second order algebraic surface given by the general equation:
- ax2 + by2 + cz2 +dxy + fyz + gx +hy + iz +j =0
- Examples of quadratic surfaces include the cone, cylinder, ellipsoid, elliptic cone, elliptic cylinder, hyperbolic cylinder, hyperbolic paraboloid, paraboloid, sphere and spheroid.

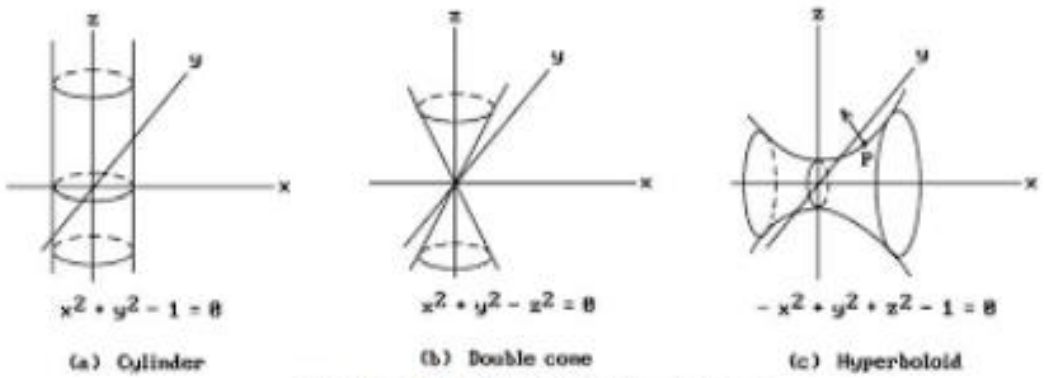


Figure 5: Quadratic Surfaces

Bezier surface

- Bezier surfaces are a species of mathematical spline used in computer graphics, computer-aided design, and finite element modeling. 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
  - the surface does not in general pass through the control points (except for the corner points)
  - the surface is contained within the convex hull of the control points.
- A two-dimensional Bézier surface can be defined as a parametric surface where the position of a point p as a function of the parametric coordinates u, v is given by:

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

- evaluated over the unit square, where

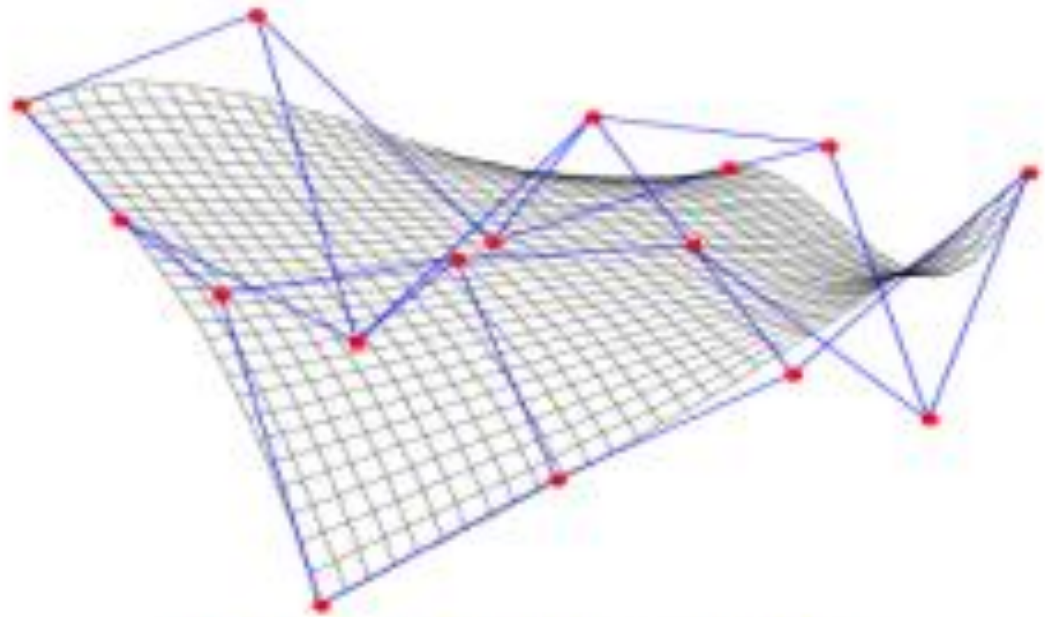
$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

- is a Bernstein polynomial, and

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

- the binomial coefficient.

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



Bézier surface