MODULE - 2

Computer Graphics & Geometric Modeling

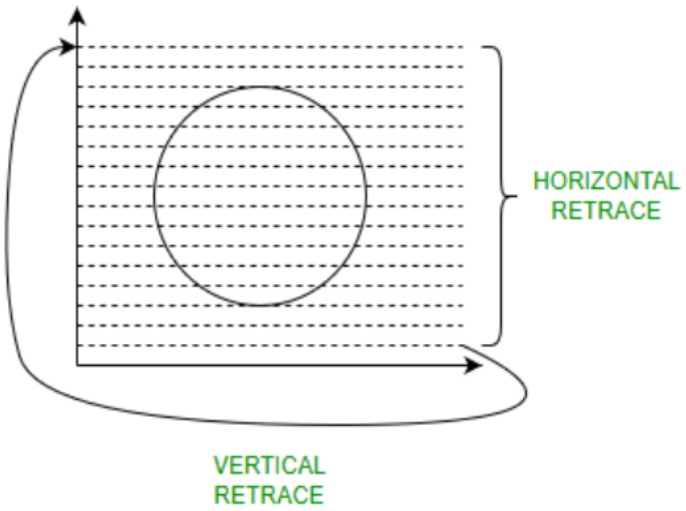
Raster Scan Graphics

- Raster scan graphics are a method of displaying images on a screen by scanning rows (lines) of pixels in a systematic order.
- Commonly used in CRT (Cathode Ray Tube) monitors and modern display devices like LCDs and LEDs.

Raster Scan Basics

- A raster is a rectangular grid of pixels (picture elements).
- Each pixel represents the smallest unit of a digital image.
- Raster scanning operates by systematically moving the electron beam or activating the display device line by line, starting from the top-left and proceeding to the bottom-right.

Raster Scan Display



Advantages of Raster Scan Graphics

- Simplicity: Easier to implement compared to vector graphics.
- Compatibility: Well-suited for modern display technologies.
- Versatility: Capable of displaying complex images, including photographs and textures.

Disadvantages of Raster Scan Graphics

- Memory Usage: Requires a large frame buffer, especially for high resolutions.
- Aliasing: Jagged edges on lines and curves due to the discrete nature of pixels.
- Limited Resolution: Image quality depends on the number of pixels (resolution).

Applications of Raster Scan Graphics

- Computer monitors and television screens.
- Video games and animations.
- Image editing and rendering software.
- CAD (Computer-Aided Design) and simulation tools.

DDA Line Algorithm (Digital Differential Analyzer)

The DDA Line Algorithm is a technique in computer graphics designed to draw a straight line connecting two points in a 2D space. It is a simple and efficient method that incrementally calculates points along the line.

Key Features

1. Incremental approach.
2. Uses floating-point arithmetic.
3. Suitable for lines with arbitrary slopes.

Steps of the DDA Algorithm

Provide the coordinates of the line's endpoints:

(x1,y1)(x_1, y_1)(x1,y1) and (x2,y2)(x_2, y_2)(x2,y2)

Compute the differences:

- **dx=x2−x1dx = x_2 - x_1dx=x2−x1**
- **dy=y2−y1dy = y_2 - y_1dy=y2−y1**

Calculate the necessary number of steps:

- Steps = $\max(|d_x|,|d_y|)$

Calculate the increment values:

- $x_{increment} = d_x/steps$
- $y_{increment} = d_y/steps$

Iteratively calculate intermediate points:

- Starting point: $(x , y)=(x_1 , y_1)$
- For each step:
  - Plot (round(x), round(y))
  - Update: $x = x + x_{increment}$, $y = y + y_{increment}$

Repeat until the endpoint is reached.

Example Problem

Draw a line from P(1,1) to Q(5,9) using DDA algorithm.

Solution: P(1,1) => x1 = 1, y1= 1

Q(5,9) => x2 = 5, y2 = 9

lets calculate the difference:

dx = x2 - x1 = 5 -1 = 4

dy = y2 - y1 = 9 -1 = 8

Length, L = abs(dy) = 8

Increment factor,

Xinc = dx/ L = 4/8 = 0.5

Yinc = dy/ L = 8/8 = 1

x=x1=1

y=y1=1

Plot(1,1)

for i=0

x = x + Xinc =1+0.5=1.5

y = y + Yinc =1+1=2

Plot(1,2)

for i=1

x = x + Xinc =1.5+0.5=2

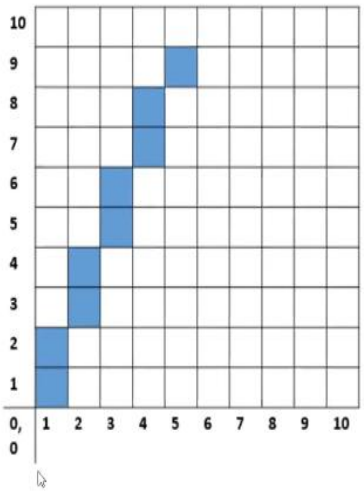
y = y + Yinc =2+1=3

Plot(2,3)

Like this we calculate others pixels

Now, list all the points to be plotted:

Steps	X	Y	X = X+ xInc	Y = Y + yInc	Plot(**int** (X), **int**(Y))
Initial	X = x1 =1	Y = y1 =1			Plot(1,1)
1	1	1	1 + 0.5 = 1.5	1 + 1 = 2	Plot(1,2)
2	1.5	2	1.5 + 0.5 = 2	2 + 1 = 3	Plot(2,3)
3	2	3	2 + 0.5 = 2.5	3 + 1 = 4	Plot(2,4)
4	2.5	4	2.5 + 0.5 = 3	4 + 1 = 5	Plot(3,5)
5	3	5	3 + 0.5 = 3.5	5 + 1 = 6	Plot(3,6)
6	3.5	6	3.5 + 0.5 = 4	6 + 1 = 7	Plot(4,7)
7	4	7	4 + 0.5 = 4.5	7 + 1 = 8	Plot(4,8)
8	4.5	8	4.5 + 0.5 = 5	8 + 1 = 9	Plot(5,9)



Advantages

- Simple to understand and implement.
- Works well for all types of lines.

Disadvantages

- Rounding operations can make it less efficient than other algorithms.
- Incremental errors can accumulate over long distances.

Applications

- Drawing lines in 2D graphics.
- Forming the basis for more advanced algorithms.

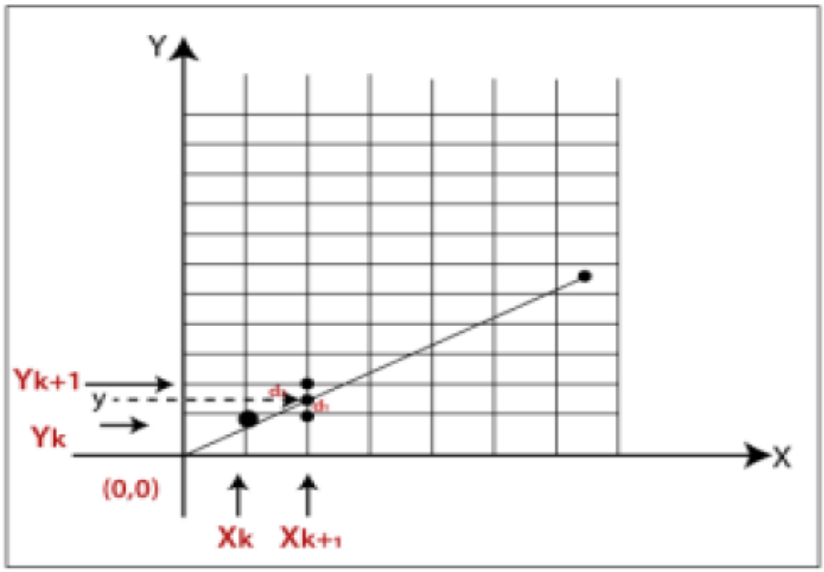
Bresenham's Line Algorithm

Bresenham's Line Algorithm is a highly efficient and accurate rasterization method for line drawing. It utilizes integer arithmetic instead of floating-point calculations, making it well-suited for digital displays.

Algorithm Concept

- It determines which pixel is closer to the theoretical line and chooses the pixel with the smallest error.
- The algorithm uses decision variables to decide between two possible pixel positions at each step.

In Bresenham’s Line Drawing Algorithm, the slope (mmm) between the starting and ending points is calculated to determine the line's trajectory.



Rephrased Steps for Bresenham’s Line Drawing Algorithm:

1. **Start.**
2. **Identify the starting point as (x1,y1)(x_1, y_1) and the ending point as (x2,y2)(x_2, y_2).**
3. **Calculate Δx\Delta x and Δy\Delta y: Δx=x2−x1,Δy=y2−y1\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1 Determine the slope: m=ΔyΔxm = \frac{\Delta y}{\Delta x}**
4. **Compute the initial decision parameter pkp_k using the formula: pk=2Δy−Δxp_k = 2\Delta y - \Delta x**
5. **Initialize the line’s starting coordinates as (xk,yk)(x_k, y_k). For the next coordinates (xk+1,yk+1)(x_{k+1}, y_{k+1}), evaluate pkp_k based on the following conditions:**
    - **Case 1: If pk<0p_k < 0: pk+1=pk+2Δy,xk+1=xk+1,yk+1=ykp_{k+1} = p_k + 2\Delta y, \quad x_{k+1} = x_k + 1, \quad y_{k+1} = y_k**
    - **Case 2: If pk≥0p_k \geq 0: pk+1=pk+2Δy−2Δx,xk+1=xk+1,yk+1=yk+1p_{k+1} = p_k + 2\Delta y - 2\Delta x, \quad x_{k+1} = x_k + 1, \quad y_{k+1} = y_k + 1**

6. **Repeat Step 5 until the endpoint of the line is reached. The total number of iterations is Δx−1\Delta x - 1.**
7. **Stop.**

Example Problem Solution:

- **Given Data:**
  Starting point: (x1,y1)=(9,18)(x_1, y_1) = (9, 18), Ending point: (x2,y2)=(14,22)(x_2, y_2) = (14, 22)
- **Step 1: Calculate Δx\Delta x and Δy\Delta y:**

Δx=14−9=5,Δy=22−18=4\Delta x = 14 - 9 = 5, \quad \Delta y = 22 - 18 = 4

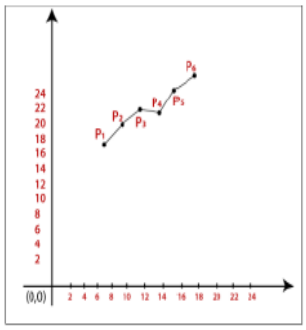
- **Step 2: Compute the decision parameter pkp_k:**

pk=2Δy−Δx=2(4)−5=3p_k = 2\Delta y - \Delta x = 2(4) - 5 = 3

- **Step 3: Evaluate cases for pkp_k:**
  - **Case 2: Since pk≥0p_k \geq 0:**
    pk+1=pk+2Δy−2Δx=3+2(4)−2(5)=1p_{k+1} = p_k + 2\Delta y - 2\Delta x = 3 + 2(4) - 2(5) = 1
    xk+1=xk+1=9+1=10,yk+1=yk+1=18+1=19x_{k+1} = x_k + 1 = 9 + 1 = 10, \quad y_{k+1} = y_k + 1 = 18 + 1 = 19
- **Step 4: Continue calculating coordinates until reaching the endpoint. Perform Δx−1=4\Delta x - 1 = 4 iterations.**

p_k	p_{k+1}	x_{k+1}	y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22

Step 5: Stop.



Coordinates of the Line:

- P1=(9,18)
- P2=(10,19)
- P3=(11,20)
- P4=(12,20)
- P5=(13,21)
- P6=(14,22)

Advantages:

- Utilizes integer arithmetic, making it faster and more efficient.
- Eliminates the need for floating-point computations.
- Effectively handles lines with all slopes.

Disadvantages:

- More complex compared to the DDA algorithm.
- Requires additional logic to manage negative slopes and steep lines.

Applications:

- Used for line rasterization in computer graphics.
- Serves as the foundation for algorithms in hardware-based graphics systems.

Co-ordinate System

A **coordinate system** is a framework used in mathematics and computer graphics to specify positions in space. It provides a structured way to describe the location of points, objects, and relationships between them.

Types of Coordinate Systems:

1. Cartesian Coordinate System (2D and 3D):
    - 2D Cartesian System: Consists of two perpendicular axes, X and Y.
    - 3D Cartesian System: Extends the 2D system by adding a third axis, Z, representing depth.

2.Polar Coordinate System: A 2D system where each point is specified by its distance from the origin and the angle from a reference axis (typically the positive X-axis).

3.Cylindrical Coordinate System (3D): A 3D system where a point is defined by a radial distance (from the origin), an angle (from the X-axis), and a height (along the Z-axis).

4.Spherical Coordinate System (3D): A 3D system where a point is defined by its radial distance, polar angle (from the Z-axis), and azimuthal angle (from the X-axis in the XY-plane).

5.Homogeneous Coordinate System: A system used in computer graphics where points in 2D or 3D space are represented using an additional dimension (usually w), facilitating easier representation of transformations such as translation.

Cartesian Coordinate System in Detail:

1.Quadrants in 2D:

- First Quadrant: $x > 0$, $y > 0$ (top-right)
- Second Quadrant: $x < 0$, $y > 0$ (top-left)
- Third Quadrant: $x < 0$, $y < 0$ (bottom-left)

- Fourth Quadrant: x > 0, y < 0 (bottom-right)

2.Axes in 3D:

- X-axis: Horizontal axis, typically representing width.
- Y-axis: Vertical axis, typically representing height.
- Z-axis: Depth axis, typically representing the forward/backward direction in 3D space.

Transformations in Coordinate Systems

Coordinate systems are essential for various transformations such as:

1. Translation
2. Scaling
3. Rotation
4. Reflection

Translation

In 2D, translation refers to shifting an object from one location to another within a two-dimensional plane.

Let's consider an object O that needs to be moved from one position to another on a 2D plane. Let:

- Initial coordinates of object O = (Xold, Yold)
- New coordinates after translation = (Xnew, Ynew)
- Translation vector (shift vector) = (Tx, Ty)

Where:

- Tx represents the distance the Xold coordinate will move.
- Ty represents the distance the Yold coordinate will move.

This translation is carried out by adding the translation vector components to the object's original coordinates:

- Xnew = Xold + Tx (Translation along the X-axis)
- Ynew = Yold + Ty (Translation along the Y-axis)

In matrix form, the translation equations can be written as:

- The homogeneous coordinates representation for (X, Y) is (X, Y, 1).
- All transformations can be applied through matrix or vector multiplications.

The translation matrix is a 3 x 3 matrix, represented as:

[10Tx01Ty001]\begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix}100010TxTy1

Example Problem

Given a circle C with a radius of 10 and center at (1, 4), apply a translation with a shift of 5 along the X-axis and 1 along the Y-axis. Find the new coordinates of C, keeping its radius unchanged.

Solution

Given:

- Old center coordinates of C = (Xold, Yold) = (1, 4)
- Translation vector = (Tx, Ty) = (5, 1)

To find the new center coordinates of C = (Xnew, Ynew):

- Xnew = Xold + Tx = 1 + 5 = 6
- Ynew = Yold + Ty = 4 + 1 = 5

Thus, the new center coordinates of C are (6, 5).

Alternatively,

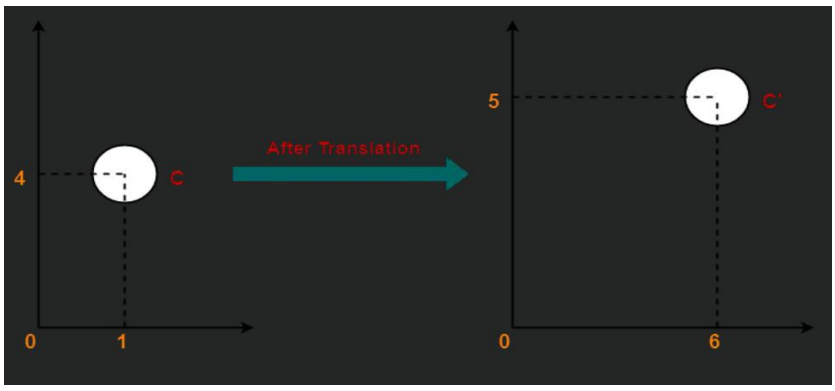
In matrix form, the new center coordinates of C after translation may be obtained as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

Thus, New center coordinates of C = (6, 5).



Scaling in Computer Graphics

Scaling in computer graphics refers to the process of adjusting the size of an object, either increasing or decreasing it.

- Scaling allows you to modify the size of an object.
- The scaling transformation changes the coordinates of the object's original points.

- The scaling factor determines whether the object will increase or decrease in size:
    - If the scaling factor is greater than 1, the object will enlarge.
    - If the scaling factor is less than 1, the object will shrink.

Consider an object O that needs to be scaled in a 2D plane.

Let:

- Initial coordinates of the object O = (Xold, Yold)
- Scaling factor along the X-axis = Sx
- Scaling factor along the Y-axis = Sy
- New coordinates of the object O after scaling = (Xnew, Ynew)

Scaling is achieved by using the following equations:

- Xnew = Xold × Sx
- Ynew = Yold × Sy

In matrix form, the scaling equations can be represented as a 3x3 matrix for homogeneous coordinates.

Example Problem

Given a square object with the corner points A(0, 3), B(3, 3), C(3, 0), and D(0, 0), apply scaling factors of 2 along the X-axis and 3 along the Y-axis to determine the new coordinates of the square.

Solution

Given:

- Initial coordinates of the square: A(0, 3), B(3, 3), C(3, 0), D(0, 0)
- Scaling factor for the X-axis = 2
- Scaling factor for the Y-axis = 3

For each point, apply the scaling equations:

For corner A(0, 3):

- Xnew = 0 × 2 = 0
- Ynew = 3 × 3 = 9

  New coordinates of A: (0, 9)

For corner B(3, 3):

- Xnew = 3 × 2 = 6
- Ynew = 3 × 3 = 9

  New coordinates of B: (6, 9)

For corner C(3, 0):

- Xnew = 3 × 2 = 6

- Ynew = 0 × 3 = 0

  New coordinates of C: (6, 0)

For corner D(0, 0):

- Xnew = 0 × 2 = 0
- Ynew = 0 × 3 = 0

  New coordinates of D: (0, 0)

Thus, the new coordinates of the square after scaling are:

A(0, 9), B(6, 9), C(6, 0), D(0, 0)

Rotation

Rotation in 2D

2D rotation is the process of rotating an object in a two-dimensional plane around a specific angle.

Consider an object O that needs to be rotated from one angle to another in the 2D plane.

Let:

- Initial coordinates of object O = (Xold, Yold)
- Initial angle of object O relative to the origin = Φ
- Rotation angle = θ
- New coordinates of object O after rotation = (Xnew, Ynew)

The rotation transformation is applied using the following equations:

- Xnew = Xold × cos(θ) – Yold × sin(θ)
- Ynew = Xold × sin(θ) + Yold × cos(θ)

In matrix form, the above rotation equations can be represented as a 3x3 matrix for homogeneous coordinates.

Example Problem

Consider a line segment with the starting point at (0, 0) and the ending point at (4, 4). We will apply a 30° counterclockwise rotation to the line segment and determine the new coordinates of the line.

Solution:

We rotate the line by applying the same angle to both endpoints and then draw a new line between the transformed endpoints.

Given:

- Old endpoint coordinates of the line = (Xold, Yold) = (4, 4)
- Rotation angle = θ = 30º

To find the new endpoint coordinates after rotation, we use the rotation equations:

- Xnew = Xold × cos(30º) – Yold × sin(30º)
  = 4 × (√3 / 2) – 4 × (1 / 2)
  = 2√3 – 2
  = 2(√3 – 1)
  = 2(1.73 – 1)
  = 1.46
- Ynew = Xold × sin(30º) + Yold × cos(30º)
  = 4 × (1 / 2) + 4 × (√3 / 2)
  = 2 + 2√3
  = 2(1 + √3)
  = 2(1 + 1.73)
  = 5.46

Thus, the new coordinates of the endpoint after rotation are (1.46, 5.46).

Alternatively, the new coordinates can also be derived using the rotation matrix.

Reflection in 2D

Reflection is a special case of rotation where the object is rotated by 180°. The reflected object appears as a mirror image on the opposite side of the axis of reflection, and its size remains unchanged.

Consider an object O that needs to be reflected in the 2D plane. Let:

- Initial coordinates of the object O = (Xold, Yold)
- New coordinates of the reflected object O = (Xnew, Ynew)

Reflection on the X-Axis:

The reflection on the X-axis follows these equations:

- Xnew = Xold
- Ynew = -Yold

In matrix form, the reflection equations can be represented as a 3x3 matrix for homogeneous coordinates.

Reflection on the Y-Axis:

The reflection on the Y-axis follows these equations:

- Xnew = -Xold
- Ynew = Yold

In matrix form, the reflection equations can be represented as a 3x3 matrix for homogeneous coordinates.

Example Problem

Consider a triangle with the corner points A(3, 4), B(6, 4), and C(5, 6). We will apply a reflection across the X-axis and determine the new coordinates of the triangle.

Solution:

Given:

- Initial corner coordinates of the triangle: A(3, 4), B(6, 4), C(5, 6)
- Reflection is applied along the X-axis

For each corner of the triangle, the reflection equations are applied:

For corner A(3, 4):

- Xnew = Xold = 3
- Ynew = -Yold = -4

  Thus, the new coordinates of corner A after reflection are (3, -4).

For corner B(6, 4):

- Xnew = Xold = 6
- Ynew = -Yold = -4

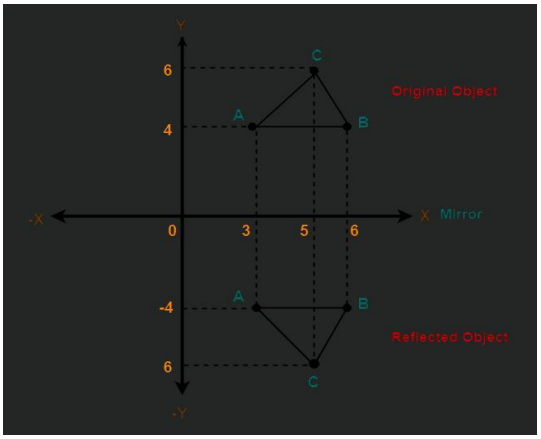
  Thus, the new coordinates of corner B after reflection are (6, -4).

For corner C(5, 6):

- Xnew = Xold = 5
- Ynew = -Yold = -6

  Thus, the new coordinates of corner C after reflection are (5, -6).

The new coordinates of the triangle after reflection are A(3, -4), B(6, -4), and C(5, -6)



Geometric Modeling

Geometric modeling is the mathematical representation of the shape and structure of objects. It is a fundamental concept in computer graphics, CAD (Computer-Aided Design), 3D animation, and engineering for designing, analyzing, and visualizing objects and environments.

Introduction

Geometric modeling is the mathematical representation of the shape and structure of objects. It is a fundamental concept in computer graphics, CAD (Computer-Aided Design), 3D animation, and engineering for designing, analyzing, and visualizing objects and environments.

Types of Geometric Models

1. Wireframe Models

Wireframe models represent objects using edges and vertices, forming a skeleton-like structure. These models do not contain surfaces, making them lightweight and efficient for visualization and early-stage design. However, they lack depth and realism.

- **Advantages**: Simple to construct, requires less memory, and allows easy modifications.
- **Disadvantages**: No surface representation, making it difficult to interpret hidden parts.
- **Applications**: Used in initial design concepts, motion analysis, and computer-aided design (CAD).

2. Surface Models

Surface models define an object using connected surfaces (e.g., triangles, quadrilaterals, or curved patches). Unlike wireframe models, surface models can provide a more realistic representation by including textures and shading.

- **Advantages**: More realistic than wireframe models, supports rendering and visualization.
- **Disadvantages**: No volume representation, making operations like mass calculations difficult.
- **Applications**: Used in gaming, animation, industrial design, and automotive industries.

3. Solid Models

Solid models define objects with volume and mass properties, making them the most comprehensive representation. These models include interior details and support physical simulations.

- **Advantages**: Provides complete object representation, supports Boolean operations, and allows engineering analysis.
- **Disadvantages**: Requires more memory and computational power.
- **Applications**: Used in CAD, 3D printing, engineering simulations, and manufacturing.

Geometric Modeling Representations

1. Polygonal Modeling

Polygonal modeling represents objects using a mesh of polygons (typically triangles or quadrilaterals). This approach is widely used in graphics and real-time rendering.

- **Key Features**: Defined by vertices, edges, and faces; easy to render using GPUs.
- **Advantages**: Efficient for real-time applications, supports hardware acceleration.
- **Disadvantages**: Requires a large number of polygons for smooth surfaces, leading to high computational costs.
- **Applications**: Used in gaming, VR, animation, and architectural visualization.

2. Spline and Curve-Based Modeling

This method uses mathematical curves (such as Bézier, B-spline, and NURBS) to define smooth surfaces and shapes.

- **Key Features**: Uses control points to manipulate curves and surfaces.
- **Advantages**: Creates smooth and precise shapes with fewer data points.
- **Disadvantages**: More complex mathematical representation and requires additional processing.
- **Applications**: Used in industrial design, automotive engineering, and animation.

3. Parametric Modeling

Parametric modeling represents objects using parameters and constraints, allowing dynamic updates when dimensions change.

- **Key Features**: Uses mathematical formulas to define relationships between elements.
- **Advantages**: Allows easy modifications, maintains design intent.
- **Disadvantages**: Requires careful constraint management, can become complex.
- **Applications**: Used in CAD software for mechanical and architectural design.

4. Implicit Modeling

Implicit modeling represents shapes using mathematical functions rather than explicit geometric definitions. For example, a sphere can be represented as x2+y2+z2−r2=0x^2 + y^2 + z^2 - r^2 = 0x2+y2+z2−r2=0.

- **Key Features**: Defines surfaces by equations instead of vertices.
- **Advantages**: Ideal for smooth organic shapes, supports advanced operations like blending.
- **Disadvantages**: Complex to render directly, requires conversion for visualization.
- **Applications**: Used in fluid simulations, volumetric rendering, and scientific visualization.

5. Voxel-Based Modeling

Voxel-based modeling represents objects using a 3D grid of small cubes (voxels), similar to pixels in 2D images.

- **Key Features**: Stores volumetric data instead of surfaces or edges.
- **Advantages**: Useful for medical imaging, easy to process for physics simulations.
- **Disadvantages**: High memory usage, limited detail compared to polygonal models.
- **Applications**: Used in medical imaging (CT scans, MRIs), scientific simulations, and voxel-based games like Minecraft.

Applications of Geometric Modeling:

- CAD software for mechanical design (e.g., AutoCAD, SolidWorks, CATIA, etc.)
- CAM for tool path generation in CNC machining
- Finite element analysis (FEA) for stress and deformation studies
- 3D printing and rapid prototyping
- Robotics and automation for path planning and object recognition
- Architectural and structural modeling for construction planning

Tools and Software

- Blender, Maya, SolidWorks, AutoCAD, CATIA, Rhino, Houdini.

Requirements of Geometric Modeling

Geometric modeling systems must satisfy several essential requirements to be effective in design and manufacturing:

Accuracy

- The model should precisely represent the physical object with minimal geometric deviation.
- Example: In aerospace engineering, exact dimensions are critical for aerodynamics and structural integrity.
- High precision reduces errors in manufacturing and ensures better product quality.

Computational Efficiency

- The model should be computationally efficient for rendering and simulation.

- Example: A CAD system should handle complex surfaces without excessive memory usage or slow performance.
- Efficient algorithms optimize processing speed, reducing the load on computer hardware.

Flexibility and Modifiability

- The model should allow modifications without the need for complete reconstruction.
- Example: Parametric modeling in SolidWorks allows easy modifications of dimensions without redrawing the object.
- Designers can quickly iterate and refine their designs, improving overall productivity.

Compatibility and Interoperability

- The model should be compatible with various CAD, CAM, and CAE software.
- Example: IGES and STEP formats allow interoperability between different CAD platforms.
- Interoperability ensures smooth data exchange between different systems and departments.

Topological and Geometric Consistency

- The model should ensure that surfaces and edges meet correctly without gaps or overlaps.
- Example: In automotive design, gaps between modeled parts can lead to assembly errors.
- Proper consistency ensures that manufactured components fit together seamlessly.

Advantages of Geometric Modeling

- Facilitates accurate and realistic representation of objects.
- Allows for easy modifications and design iterations.
- Enhances visualization, reducing the need for physical prototypes.
- Improves manufacturing efficiency and reduces production costs.
- Essential for automation and simulation in modern industries.

Limitations of Geometric Modeling

- High computational power is required for complex models.
- Steep learning curve for advanced CAD/CAM software.
- Software and hardware costs can be significant.
- Errors in modeling can lead to manufacturing defects if not carefully analyzed.

Importance of Geometric Modeling in Engineering and Manufacturing

- Enables engineers to create precise and functional designs.

- Supports optimization and analysis of structural integrity.
- Reduces the risk of design flaws through virtual testing.
- Plays a crucial role in industries like aerospace, automotive, and biomedical engineering.
- Integrates with manufacturing processes for CNC machining and additive manufacturing.

Representations in Geometric Modelling

- Parametric Representation
- Non-Parametric Representation

Parametric Representation

Parametric representation is a mathematical method used to describe geometric objects through parameters. In this approach, the coordinates of points on a curve or surface are **expressed as functions of one or more independent parameters.** This representation is commonly used in computer graphics, CAD, and geometric modeling to define curves, surfaces, and shapes.

Parametric Representation of Curves and Surfaces

Definition of Parametric Representation

In parametric representation, geometric entities are expressed as mathematical functions of one or more parameters. This representation provides greater control over complex geometries and allows designers to manipulate and modify shapes easily.

Parametric Equations for Curves

A parametric curve in 3D space is defined as:

t = parameter (scalar variable)

$x = f(t), y = g(t), z = h(t)$ (where f, g, h are continuous functions)

Example: Parametric Equation of a Circle

For a circle of radius R centered at the origin:

$x = R \cos(t)$ $y = R \sin(t)$ $(0 \leq t \leq 2\pi)$

Advantages of Parametric Representation

- Provides better control over complex shapes.
- Allows smooth transformations and animations.
- Used in CAD software for defining splines, Bézier curves, and B-splines.
- More efficient for modeling and rendering complex geometries.
- Enables better integration with design constraints and optimization techniques.

Parametric Surfaces

A parametric surface is represented as:

$x = f(u,v)$, $y = g(u,v)$, $z = h(u,v)$ (where u, v are independent parameters)

Example: Parametric Equation of a Sphere

A sphere of radius R centered at the origin is given by:

$x = R \cos(u) \sin(v)$ $y = R \sin(u) \sin(v)$ $z = R \cos(v)$ $(0 \leq u \leq 2\pi, 0 \leq v \leq \pi)$

Non-Parametric Representation of Curves and Surfaces

Definition of Non-Parametric Representation

In non-parametric representation, geometric entities are expressed in implicit or explicit mathematical forms rather than as functions of parameters. This approach is commonly used for simpler geometric shapes where direct equations are more suitable.

Implicit Representation

- A geometric shape is defined as a solution to an equation.
- Example: A circle of radius R is defined implicitly as:

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

Explicit Representation

- The geometric entity is defined as an explicit function of independent variables.
- Example: A straight line in 2D:

$y = mx + c$

Advantages of Non-Parametric Representation

- Simpler mathematical representation for basic shapes.
- Efficient for standard geometries like lines, planes, and circles.
- Useful in applications where objects do not require complex modifications.

Disadvantages of Non-Parametric Representation

- Limited in handling complex and freeform shapes.
- Not suitable for advanced CAD modeling applications.
- Less flexible in terms of modifications and transformations

Curve Representation

Curve representation is a critical concept in computer graphics, geometric modeling, and CAD (Computer-Aided Design). Curves are fundamental geometric entities that can be defined and manipulated in various ways, depending on the requirements of the application. Effective curve representation allows for smooth transitions, complex shapes, and efficient rendering.

Comparison of Curve Representation Methods

Method	Definition	Advantages	Disadvantages
Parametric	Coordinates expressed as functions of a parameter	Flexible and detailed	Can be complex for higher-order curves
Implicit	Defined by an implicit equation	Simple for intersections	Less intuitive for control
Explicit	Direct relationship between variables	Straightforward for simple curves	Limited flexibility for complex shapes
Piecewise Linear	Series of connected line segments	Simple and efficient	Not smooth; can be visually less appealing
Bézier	Defined by control points	Intuitive control; good for animations	Limited local control for higher-order
B-Splines	Generalized Bézier curves	High flexibility; local control	More complex to compute
NURBS	Rational curves with weights	atile; can represent complex shapes	More complex representation and manipulation

Analytical Curve Representation

Definition

Analytical curve representation uses mathematical equations to describe curves precisely. The curves are defined by explicit mathematical formulas that specify the relationship between coordinates in a Cartesian coordinate system.

Characteristics

- **Mathematical Expressions:** Curves are expressed using mathematical functions or equations.
- **Exact Representation:** The shape of the curve can be computed exactly for any given point.

Types of Equations:

    - Polynomial Equations
    - Trigonometric Functions
    - Implicit Functions

Advantages

- **Precision:** Allows for precise calculations and exact definitions of curves.
- **Ease of Analysis:** Simplifies mathematical analysis and derivation of properties (e.g., curvature, intersections).
- **Wide Applicability:** Suitable for a variety of shapes and applications.

Disadvantages

- **Complexity for Higher-Order Curves:** As the degree of the polynomial increases, the representation and computations can become complex.
- **Limited Control:** Modifying the shape often requires changing the parameters of the equation rather than directly manipulating the curve.

Applications

- Used in fields requiring precise geometric definitions, such as robotics, engineering simulations, and mathematical modeling.

Synthetic Curve Representation

Definition

Synthetic curve representation focuses on creating curves through geometric techniques or control points, rather than relying on mathematical equations. This method often involves the use of control points, weights, and specialized algorithms to generate the desired curves..

Characteristics

- Control Points: A set of control points is used to define the curve, influencing its shape and trajectory.
- Constructive Methods: Curves are created through geometric techniques like interpolation or approximation.

Types of Synthetic Representations:

- Bézier Curves
- B-Splines
- NURBS (Non-Uniform Rational B-Splines)

Advantages:

- Intuitive Control: Direct manipulation of control points allows for easy design and adjustment of curves.
- Local Control: Modifying a control point affects only a segment of the curve, offering precise fine-tuning.
- Smoothness: Synthetic representations ensure smooth transitions between curve segments.

Disadvantages:

- Less Exact: The curve may not match the control points exactly, especially with Bézier curves and splines.
- Computational Complexity: Some synthetic methods may demand more computational resources for rendering and evaluation.

Applications:

- Widely used in computer graphics, animation, CAD, and design applications, where flexibility and user control are crucial.

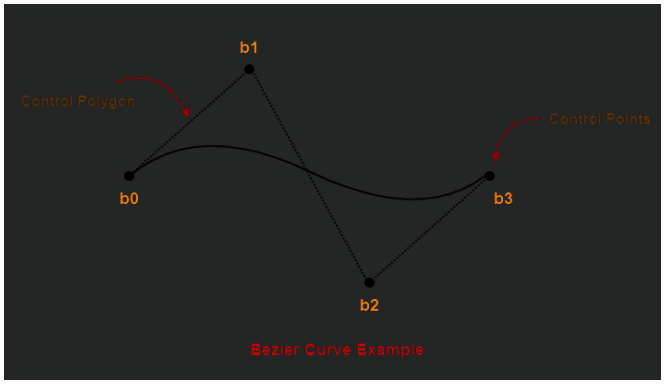
Bézier Curves

Bézier curves are parametric curves extensively used in computer graphics, animation, and CAD (Computer-Aided Design) to model smooth curves. They are defined by a set of control points that determine the shape and behavior of the curve. Developed by Pierre Bézier in the 1960s, Bézier curves are known for their intuitive control and computational efficiency.

A Bézier curve can be defined as:

- A parametric curve defined by a set of control points.
- The two endpoints of the curve are the first and last control points.
- The intermediate control points determine the shape of the curve.

The concept of bezier curves was given by Pierre Bezier.



Here,

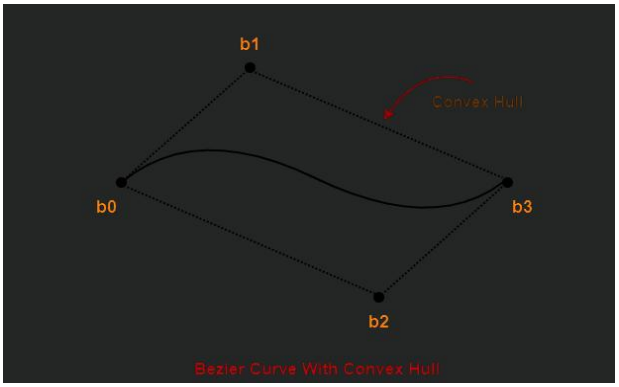
- This bezier curve is defined by a set of control points b_0, b_1, b_2 and b_3.
- Points b_0 and b_3 are ends of the curve.
- Points b_1 and b_2 determine the shape of the curve.

Bezier Curve Properties-

Few important properties of a bezier curve are-

Property-01:

Bezier curve is always contained within a polygon called as convex hull of its control points.



Property-02:

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

Property-03:

The degree of the polynomial defining the curve segment is one less than the total number of control points.

Degree = Number of Control Points – 1

Property-04:

The order of the polynomial defining the curve segment is equal to the total number of control points.

Order = Number of Control Points

Property-05:

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

Bezier Curve Equation-

A bezier curve is parametrically represented by-

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

Bezier Curve Equation

Here,

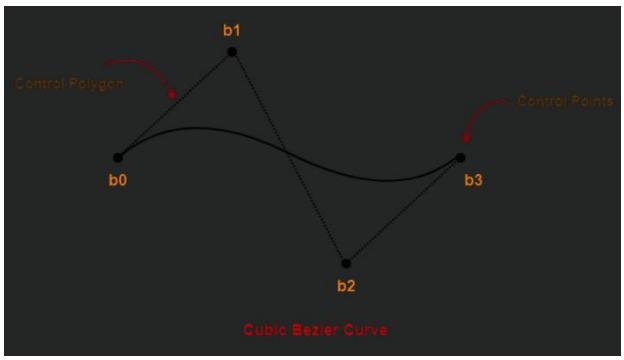
- t is any parameter where 0 <= t <= 1
- P(t) = Any point lying on the bezier curve
- B_i = i^{th} control point of the bezier curve
- n = degree of the curve
- $J_{n,i}(t)$ = Blending function = $C(n , i)\ t^i\ (1 - t)^{n-i}$ where $C(n , i) = n! / i!(n - i)!$

Cubic Bezier Curve-

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

Example

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



Here,

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

Cubic Bezier Curve Equation-

The parametric equation of a bezier curve is-

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

Bezier Curve Equation

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

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

Expanding the above equation, we get-

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

Now,

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

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

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

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

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

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

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

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

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

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

Problem:

Given a bezier curve with 4 control points

B_0 [1 0] , B_1[3 3] , B_2[6 3] , B_3[8 1]

Determine any 5 points lying on the curve. Also, draw a rough sketch of the curve.

Solution

We have

- The given curve is defined by 4 control points.
- So, the given curve is a cubic bezier curve.

The parametric equation for a cubic bezier curve is,

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

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

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

Now,

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

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

For t = 0:

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

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

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

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

For t = 0.2:

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

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

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

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

$P(0.2) = [1\ 0] \times 0.512 + [3\ 3] \times 0.384 + [6\ 3] \times 0.096 + [8\ 1] \times 0.008$

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

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

For t = 0.5:

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

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

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

$P(0.5) = [1\ 0] \times 0.125 + [3\ 3] \times 3 \times 0.5 \times 0.25 + [6\ 3] \times 3 \times 0.25 \times 0.5 + [8\ 1] \times 0.125$

$P(0.5) = [1\ 0] \times 0.125 + [3\ 3] \times 0.375 + [6\ 3] \times 0.375 + [8\ 1] \times 0.125$

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

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

For t = 0.7:

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

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

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

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

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

$P(0.7) = [1\ 0] \times 0.027 + [3\ 3] \times 0.189 + [6\ 3] \times 0.441 + [8\ 1] \times 0.343$

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

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

For t = 1:

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

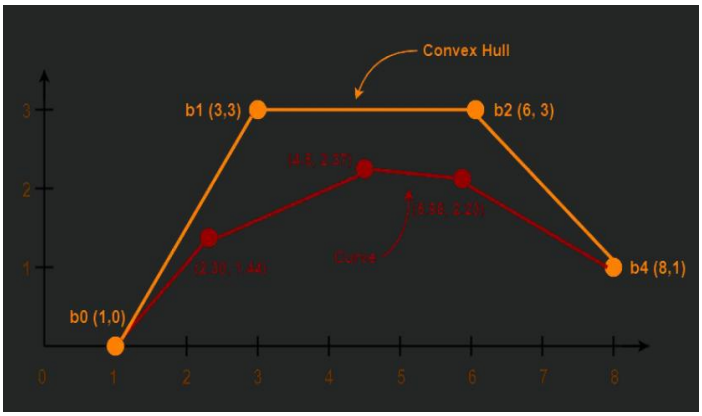
$P(1) = [1\ 0](1-1)^3 + [3\ 3]3(1)(1-1)^2 + [6\ 3]3(1)^2(1-1) + [8\ 1](1)^3$

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

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

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

Following is the required rough sketch of the curve,

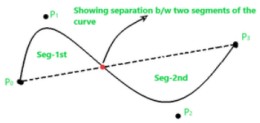


B-Splines

B-spline curves were introduced to address the limitations of Bézier curves. While both curves are parametric, Bézier curves exhibit a significant drawback: when a control point is moved, the entire shape of the curve changes. In contrast, with B-spline curves, adjusting a control point only affects a specific segment of the curve, rather than altering the entire shape.

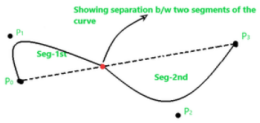
In B-splines, the control points provide local control over the curve's shape, as opposed to the global control seen in Bézier curves. This allows for more flexibility and precision when modifying the curve.

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



Control points { P0, P1, P2, P3}

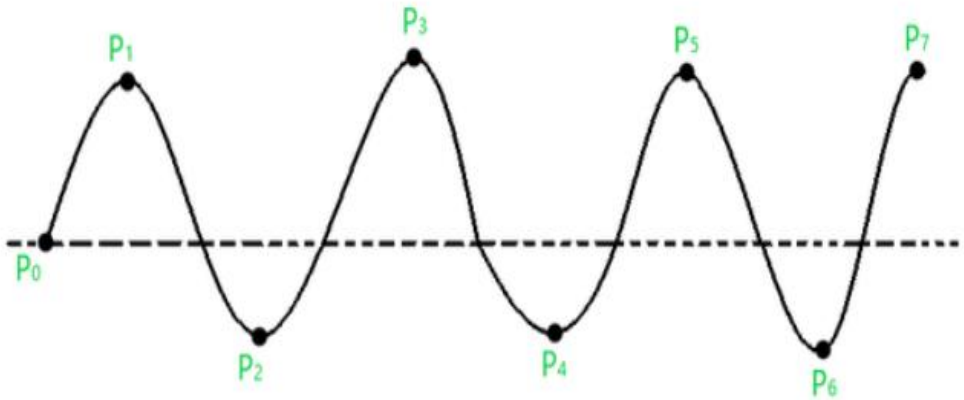
B-spline curve shape after changing the position of control point P_1 –



Control points { P0, P1, P2, P3}

B-spline Curve:

As mentioned earlier, B-spline curves are independent of the number of control points and are formed by smoothly joining several segments. Each segment's shape is determined by a specific set of control points that influence that region of the segment. The curve is constructed by combining these segments, resulting in a smooth, continuous curve. Below is an example of a B-spline curve, where each segment is influenced by its corresponding control points.



Properties of B-spline Curve:

- Each basis function has a non-negative value for all parameters.
- Every basis function has a single maximum value, except when k=1k = 1k=1.

- The degree of a B-spline curve polynomial is independent of the number of control points, which enhances its reliability over Bézier curves.
- B-spline curves provide local control via the control points, influencing each segment of the curve individually.
- The sum of the basis functions for any given parameter is equal to one.

NURBS (Non-Uniform Rational B-Splines):

NURBS (Non-Uniform Rational B-Splines) are an advanced mathematical representation used for modeling curves and surfaces. They offer a high level of flexibility and accuracy in creating complex shapes. NURBS extend B-splines by incorporating weights, enabling the representation of both standard geometric shapes (like circles and ellipses) and more intricate, freeform shapes. NURBS are widely used in computer graphics, CAD (Computer-Aided Design), and geometric modeling due to their precision and versatility in handling diverse modeling tasks.