2D Coordinate System

1. Introduction

A coordinate system is a reference system that uses one or more numbers (coordinates) to uniquely identify the position of a point in a plane. In computer graphics, 2D coordinate system is used to represent and locate points, lines, shapes and objects on a 2D display device.

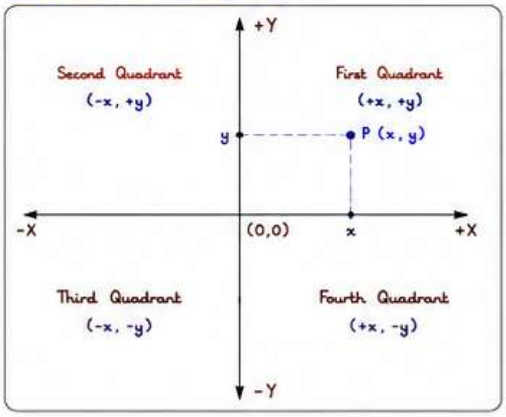
2. Definition

A 2D coordinate system is a system in which any point on a plane is specified by an ordered pair (x, y) of real numbers, where 'x' represents the horizontal position (abscissa) and 'y' represents the vertical position (ordinate) of the point.

3. Basic Concepts

- The plane is divided into four regions called quadrants.
- The two perpendicular axes intersect at origin (0, 0).
- The horizontal axis is called X-axis.
- The vertical axis is called Y-axis.
- The point (x, y) is located x units along X-axis and y units along Y-axis from the origin.

4. 2D Coordinate System



Key Point :
Any point P in 2D plane is represented by an ordered pair (x, y).
If P is in:

- First Quadrant → $x > 0,\ y > 0$
- Second Quadrant → $x < 0,\ y > 0$
- Third Quadrant → $x < 0,\ y < 0$
- Fourth Quadrant → $x > 0,\ y < 0$

5. Coordinates of Some Special Points

- Origin : (0, 0)
- On +X axis : (x, 0)
- On -X axis : (-x, 0)
- On +Y axis : (0, y)
- On -Y axis : (0, -y)
- In general : (x, y)

6. Distance Between Two Points

Let two points be $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. The distance between P_1 and P_2 is given by

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

7. Midpoint Formula

The midpoint M of a line segment joining $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ is

$$M\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

8. Example

Find the distance and midpoint between $P_1(2, 3)$ and $P_2(8, 7)$.

Solution :

Distance,

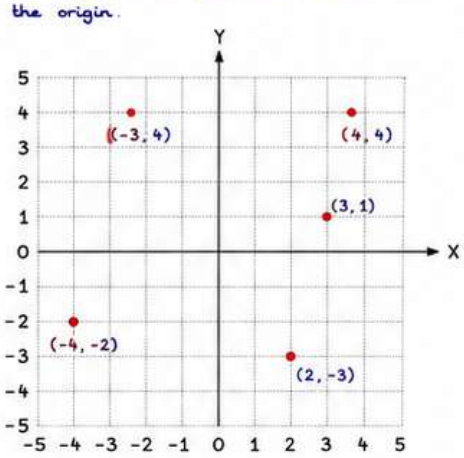
$$d = \sqrt{(8 - 2)^2 + (7 - 3)^2}$$
$$= \sqrt{6^2 + 4^2} = \sqrt{36 + 16}$$
$$= \sqrt{52} = 2\sqrt{13} \text{ units}$$

Midpoint,

$$M\left(\frac{2 + 8}{2}, \frac{3 + 7}{2}\right)$$
$$= M\left(\frac{10}{2}, \frac{10}{2}\right)$$
$$= M(5, 5)$$

9. Representation of Points

Points can be plotted on the 2D plane using their coordinates. The position of any point is fixed with respect to the origin.



10. Applications

- Used to display 2D objects on monitor screen.
- Used in drawing basic shapes like lines, circles, rectangles, polygons.
- Used in GUI design, animation, games, CAD systems.
- Used for tranformations like translation, rotation, scaling, reflection.

11. Advantages

- ✓ Simple and easy to understand.
- ✓ Requires less memory.
- ✓ Efficient for 2D graphics operations.
- ✓ Forms the basis for 3D graphics.

12. Limitations

- ✗ Cannot represent 3D objects.
- ✗ Limited to two dimensions only.
- ✗ Not suitable for complex 3D scenes.

Important Points

- Origin (0, 0) is the intersection of X and Y axes.
- A point in 2D is represented as (x, y).
- Four quadrants are formed.
- Distance and midpoint formulas are important.
- Basis for all 2D tranformations and graphics.

Typical Exam Questions

1. Define 2D coordinate system.
2. Draw 2D coordinate system and label quadrants.
3. Find distance between (2, 3) and (6, 7).
4. Find midpoint between (4, 1) and (10, 5).
5. Write the coordinates of points on axes.

Summary

2D coordinate system uses two perpendicular axes X and Y to locate any point in a plane using an ordered pair (x, y). It is the basic building block of computer graphics and is used for plotting points, drawing shapes and applying tranformations.

3D Coordinate System

1. Introduction

A 3D coordinate system is a reference system in which any point in space is represented by an ordered triplet (x, y, z) of real numbers.
It has three mutually perpendicular axes: X, Y and Z.

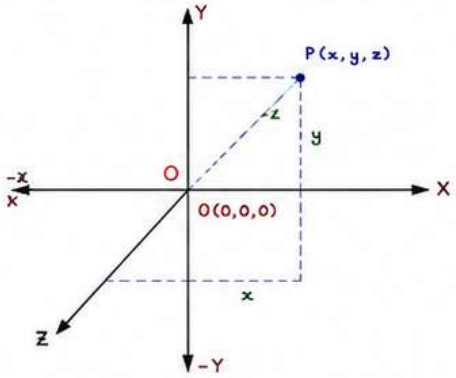
2. Definition

A 3D coordinate system is a system in which the position of any point in space is specified by its perpendicular distances from three mutually perpendicular planes.
These distances are called x-coordinate, y-coordinate and z-coordinate.

3. Axes and Planes

- X-axis : Horizontal axis (Left - Right)
- Y-axis : Vertical axis (Up - Down)
- Z-axis : Depth axis (Front - Back)
- XY-plane : Horizontal plane (z = 0)
- YZ-plane : Vertical plane (x = 0)
- ZX-plane : Vertical plane (y = 0)
- All three axes intersect at the origin (0,0,0).

4. 3D Coordinate System Diagram



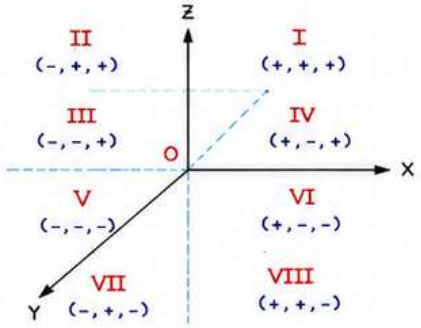
Note :
- x is the distance from YZ-plane.
- y is the distance from ZX-plane.
- z is the distance from XY-plane.

5. Coordinates of Some Important Points

Point	Coordinates (x, y, z)
Origin	(0, 0, 0)
On +X axis	(x, 0, 0)
On -X axis	(-x, 0, 0)
On +Y axis	(0, y, 0)
On -Y axis	(0, -y, 0)
On +Z axis	(0, 0, z)
On -Z axis	(0, 0, -z)
In general	(x, y, z)

6. Octants

The 3D space is divided by three coordinate planes (XY, YZ and ZX) into eight regions called octants.
The octants are numbered as shown below.



7. Distance Between Two Points

Let $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ be two points in space.
The distance between P_1 and P_2 is given by:

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

8. Midpoint Formula

Midpoint M between $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ is:

$$M\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2}\right)$$

9. Example

Find the distance and midpoint between $P_1(2, 1, 3)$ and $P_2(-1, 4, 7)$.

Solution :

Distance,

$$d = \sqrt{(-1-2)^2 + (4-1)^2 + (7-3)^2}$$
$$= \sqrt{(-3)^2 + (3)^2 + (4)^2}$$
$$= \sqrt{9 + 9 + 16} = \sqrt{34} \text{ units}$$

Midpoint,

$$M\left(\frac{2 + (-1)}{2}, \frac{1 + 4}{2}, \frac{3 + 7}{2}\right)$$
$$= M\left(\frac{1}{2}, \frac{5}{2}, 5\right)$$
$$= M(0.5, 2.5, 5)$$

10. Applications

- Used to represent 3D objects in computer graphics.
- Used in CAD/CAM, animation, gaming, scientific visualization.
- Used in real-time simulations, robotics, and virtual reality.
- Basic for performing 3D transformations like translation, rotation, scaling.

11. Advantages

- ✓ Helps in representing any point or object in 3D space precisely.
- ✓ Easy to perform mathematical operations and transformations.
- ✓ Essential for realistic 3D graphics and modeling.

12. Limitations

- ✗ More complex than 2D system.
- ✗ Requires more computation and memory.
- ✗ Difficult to visualize without projection on 2D screen.

Important Points

- 3D coordinate system has three mutually perpendicular axes: X, Y and Z.
- Any point in space is represented by (x, y, z).
- Space is divided into 8 octants by coordinate planes.
- Distance and midpoint formulas are very important.

Typical Exam Questions

1. Define 3D coordinate system and draw diagram.
2. Write coordinates of important points in 3D space.
3. Find distance between (1, 2, 3) and (4, 6, 8).
4. Find midpoint between (2, -1, 3) and (6, 3, 7).
5. Explain octants in 3D coordinate system.

Summary

3D coordinate system is used to locate and represent points in three-dimensional space. It is the foundation for all 3D graphics operations including transformations, projections, clipping and rendering.

2D Translation

1. Introduction

Translation is a basic transformation in which every point of an object is shifted (moved) from its original position by a fixed distance in a specified direction along the X-axis, Y-axis or both.

2. Definition

If a point P(x, y) is translated by tx units along X-axis and ty units along Y-axis, its new position P' (x', y') is given by

$$x' = x + tx \quad , \quad y' = y + ty$$

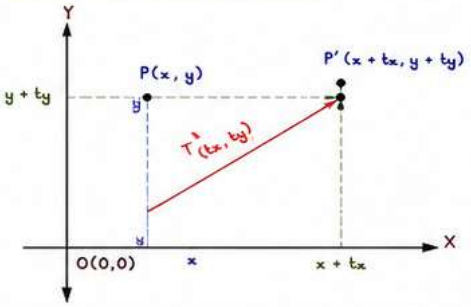
3. Translation Vector

The translation can also be represented by a translation vector.

$$T = (tx , ty)$$

New position = Old position + T

4. Geometrical Representation



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point P(x, y) is written as $[x \quad y \quad 1]^T$

The matrix form of 2D translation is

$$P' = T(tx , ty) \cdot P$$

Where,

$$T(tx , ty) = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

6. Derivation of Matrix

Let $P(x, y) \rightarrow P'(x + tx, y + ty)$

In homogeneous form,

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \\ 1 \end{bmatrix}$$

Let $T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$

Then,

$$P' = T \cdot P$$

$$\begin{bmatrix} x + tx \\ y + ty \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \quad\quad 1 \end{bmatrix}$$

For equivalence,

$$ax + by + c = x + tx$$
$$dx + ey + f = y + ty$$

Comparing coefficients,

$$a = 1, \quad b = 0, \quad c = tx$$
$$d = 0, \quad e = 1, \quad f = ty$$

So,

$$T(tx , ty) = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

7. Example

Translate the point P(2, 3) by tx = 4 units and ty = -2 units.

Solution :

Using formula,

$$x' = x + tx = 2 + 4 = 6$$
$$y' = y + ty = 3 + (-2) = 1$$

So, P' (6, 1)

Using matrix,

$$P = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} , \quad T = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = T \cdot P = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 6 \\ 1 \\ 1 \end{bmatrix} \Rightarrow P'(6, 1)$$

8. Translation in Different Directions

Direction	tx	ty	Effect
Right	+ve	0	Moves right
Left	-ve	0	Moves left
Up	0	+ve	Moves up
Down	0	-ve	Moves down
Right & Up	+ve	+ve	Moves right and up
Left & Down	-ve	-ve	Moves left and down
Right & Down	+ve	-ve	Moves right and down
Left & Up	-ve	+ve	Moves left and up

9. Properties

- Shape and size of the object do not change.
- Only position of the object changes.
- Parallel lines remain parallel after translation.
- Translation is a rigid body transformation.
- It is a basic and most used transformation in computer graphics.

10. Applications

- Used in moving objects or shapes on screen.
- Used in GUI animations and games.
- Used in scrolling of images and windows.
- Used in CAD/CAM and graphical editors.

11. Advantages

- ✓ Very simple to implement.
- ✓ Requires less computation.
- ✓ Useful for positioning and animation.
- ✓ Preserves shape, size and orientation.

12. Limitations

- ✗ Does not change size or orientation.
- ✗ Alone translation cannot create complex effects.
- ✗ Not useful for perspective or projection.

13. Important Points

- Formula : $x' = x + tx$, $y' = y + ty$
- Matrix : $T(tx , ty) = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$
- Requires 3 × 3 matrix in homogeneous form.
- Translation vector : T = (tx , ty)

Typical Exam Questions

1. Define 2D translation with example.
2. Derive the matrix for 2D translation.
3. Translate the point (3, 5) by (2, -4) using matrix method.
4. Write properties and applications of translation.

Summary

2D translation shifts every point of an object by tx units along X-axis and ty units along Y-axis. It is represented by the translation vector (tx , ty) or by a 3 × 3 translation matrix in homogeneous coordinates. Shape and size remain unchanged, only position changes.

RGPV Notes Hub

IT601 – Computer Graphics & Multimedia

Topic – 4
2D Rotation

Unit – III

2D Rotation

1. Introduction

Rotation is a transformation in which every point of an object is rotated about a fixed reference point (usually origin) through a specified angle θ.

2. Definition

If a point $P(x, y)$ is rotated about the origin by an angle θ (counter-clockwise positive), the new position $P'(x', y')$ is given by

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

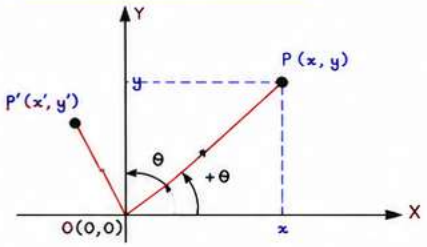
3. Rotation Vector

Rotation can also be represented by a rotation vector θ (angle of rotation).

Positive θ → Counter-clockwise rotation

Negative θ → Clockwise rotation

4. Geometrical Representation



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point $P(x, y)$ is written as

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The matrix form of 2D rotation is

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The new position is

$$P' = R(\theta) \cdot P$$

6. Derivation of Rotation Matrix

From the geometrical diagram,

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

In matrix form,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore, $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

7. Examples

(i) Rotation of $P(3, 4)$ by $90°$ counter-clockwise

$\cos 90° = 0$, $\sin 90° = 1$

$$R(90°) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

$$P' = R(90°) \cdot P = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 3 \\ 1 \end{bmatrix} \Rightarrow P'(-4, 3)$$

(ii) Rotation of $P(2, 1)$ by $180°$

$\cos 180° = -1$, $\sin 180° = 0$

$$R(180°) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

$$P' = R(180°) \cdot P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix} \Rightarrow P'(-2, -1)$$

8. Special Cases

Angle (θ)	$\cos\theta$	$\sin\theta$	Rotation Matrix $R(\theta)$
$0°$	1	0	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$90°$	0	1	$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$180°$	−1	0	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$270°$	0	−1	$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

9. Properties

- Rotation preserves the size and shape of object.
- It only changes the position and orientation.
- The distance between any two points remains unchanged.
- Parallel lines remain parallel after rotation.
- $R(-\theta) = R(\theta)^{-1} = R(\theta)^{T}$
- Determinant of $R(\theta) = 1$

10. Applications

- Used in rotating objects in computer graphics.
- Used in animation, games and simulations.
- Used in CAD/CAM and robot motion.
- Used in image processing (rotation of images).

11. Advantages

- ✓ Easy to implement using matrix multiplication.
- ✓ Requires less memory.
- ✓ Efficient and fast for real-time applications.
- ✓ Widely used in 2D graphics and transformation.

12. Limitations

- ✗ Rotation is usually about a fixed point (origin).
- ✗ For rotation about any other point, translation to origin and back is required.
- ✗ Not suitable for changing the size or shape.

13. Important Points

- Formulae : $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$
- Matrix : $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Positive θ → Counter-clockwise rotation
- Negative θ → Clockwise rotation

Typical Exam Questions

1. Define 2D rotation.
2. Derive the rotation matrix about the origin.
3. Find the coordinates of a point $(4, 2)$ rotated by $90°$ counter-clockwise.
4. Write the properties and applications of 2D rotation.

Summary

2D rotation is the transformation of an object by rotating every point about the origin through an angle θ. The new coordinates are $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$.

In homogeneous form, $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation preserves shape and size, only orientation and position change.

RGPV Notes Hub

Keep Learning, Keep Growing !

RGPV Notes Hub

IT601 – Computer Graphics & Multimedia

Unit – III

Topic – 5
2D Scaling

2D Scaling

1. Introduction

Scaling (or Zooming) is a transformation in which the size of an object is increased or decreased with respect to a fixed reference point (usually origin) by scale factors along X-axis and Y-axis.

2. Definition

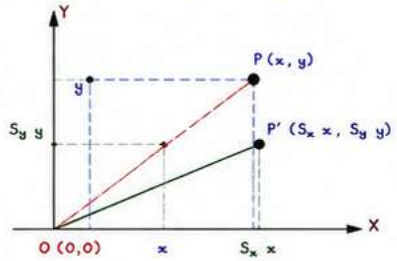
If a point $P(x, y)$ is scaled by factors S_x along X-axis and S_y along Y-axis, the new position $P'(x', y')$ is given by

$$x' = S_x \, x$$
$$y' = S_y \, y$$

3. Scale Factors

- $S_x > 1$ → Enlargement along X-axis
- $0 < S_x < 1$ → Reduction along X-axis
- $S_x = 1$ → No change along X-axis
- $S_y > 1$ → Enlargement along Y-axis
- $0 < S_y < 1$ → Reduction along Y-axis
- $S_y = 1$ → No change along Y-axis

4. Geometrical Representation



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point $P(x, y)$ is written as :

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The matrix form of 2D scaling is

$$S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The new position is

$$P' = S(S_x, S_y) \cdot P$$

6. Derivation of Scaling Matrix

Given, $x' = S_x \, x$ and $y' = S_y \, y$

In homogeneous form,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore, $S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

7. Examples

(i) Scaling by $S_x = 2$, $S_y = 3$

Matrix,

$$S(2, 3) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let $P(2, 1)$

$$P = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

$$P' = S(2,3) \cdot P = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} \Rightarrow P'(4, 3)$$

(ii) Scaling by $S_x = \frac{1}{2}$, $S_y = \frac{1}{2}$ (Reduction)

Matrix,

$$S\left(\frac{1}{2}, \frac{1}{2}\right) = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let $P(4, 6)$

$$P = \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix}$$

$$P' = S\left(\frac{1}{2}, \frac{1}{2}\right) \cdot P = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \Rightarrow P'(2, 3)$$

8. Types of Scaling

Type	Scale Factors (S_x, S_y)	Effect
Uniform Scaling	$S_x = S_y = k$	Same scale in all directions.
Non-uniform Scaling	$S_x \neq S_y$	Different scale along X and Y axes.
Enlargement	$S_x > 1$, $S_y > 1$	Object size increases.
Reduction	$0 < S_x < 1$, $0 < S_y < 1$	Object size decreases.
Reflection by Scaling	$S_x < 0$ or $S_y < 0$	Object is flipped (reflection) along axis.

9. Properties

- Scaling is done with respect to a fixed point (usually origin).
- It changes the size but not the shape of the object.
- Parallel lines remain parallel after scaling.
- Determinant of scaling matrix = $S_x \times S_y$.
- Area of object is scaled by $|S_x \times S_y|$ times.

10. Applications

- Used for zooming in / out of images.
- Used in CAD/CAM, animation and games.
- Used for resizing windows and objects in GUI.
- Used in data visualization and simulations.

11. Advantages

- ✓ Very simple mathematical operation.
- ✓ Efficient and fast to compute.
- ✓ Easy to implement using matrix multiplication.
- ✓ Useful in real-time graphics applications.

12. Limitations

- ✗ Scaling about a point other than origin requires translation before and after scaling.
- ✗ Does not preserve distances.
- ✗ Not useful for perspective view.

13. Important Points

- Formula : $x' = S_x \, x$, $y' = S_y \, y$
- Matrix : $S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- If $S_x = S_y = 1$ → No change.
- Origin remains fixed after scaling.

Typical Exam Questions

1. Define 2D scaling.
2. Derive the scaling matrix.
3. Scale the point (3, 4) by factors (2, ½).
4. Write properties and applications of 2D scaling.

Summary

2D scaling is used to increase or decrease the size of an object with respect to a fixed point (usually origin) by scale factors S_x and S_y along X and Y axes. It is represented by the scaling matrix $S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ in homogeneous coordinates. It preserves shape and orientation but not size.

RGPV Notes Hub

Keep Learning, Keep Growing !

2D Reflection

1. Introduction

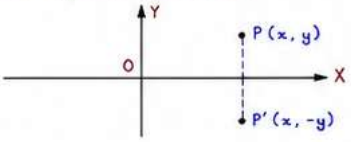
Reflection is a transformation that produces a mirror image of an object with respect to a reference line (axis).

2. Definition

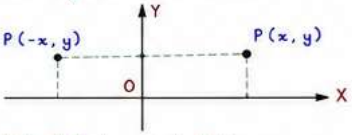
If a point $P(x, y)$ is reflected about a line (axis), the new position $P'(x', y')$ is obtained by changing the sign of the coordinate(s) according to the axis.

3. Geometrical Representation

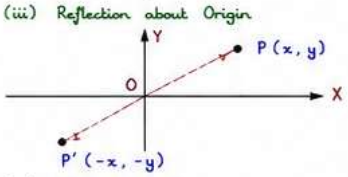
(i) Reflection about X-axis



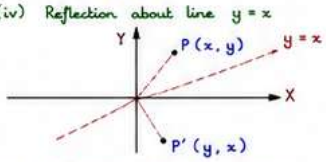
(ii) Reflection about Y-axis



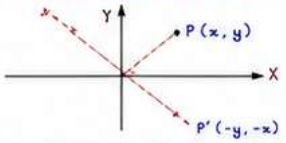
(iii) Reflection about Origin



(iv) Reflection about line $y = x$



(v) Reflection about line $y = -x$



Important Points

- Reflection is a mirror image of the object.
- Only the position and orientation change.
- Size and shape of the object remain same.
- Reflection is its own inverse.
- Determinant of reflection matrix = -1.

4. Reflection Formulas

Axis / Line	New Coordinates (x', y')
About X-axis	$(x, -y)$
About Y-axis	$(-x, y)$
About Origin	$(-x, -y)$
About line $y = x$	(y, x)
About line $y = -x$	$(-y, -x)$

5. Homogeneous Coordinates

In homogeneous coordinates,

$$P(x, y) \rightarrow P \begin{bmatrix} x & y & 1 \end{bmatrix}^T$$

6. Reflection Matrix

The reflection transformation can be represented by a 3×3 matrix.

Axis / Line of Reflection	Reflection Matrix R
About X-axis	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
About Y-axis	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
About Origin	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
About line $y = x$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
About line $y = -x$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

7. General Form

If R is the reflection matrix, then

$$P' = R \cdot P$$

where P and P' are position vectors in homogeneous coordinates.

8. Examples

(i) Reflect $P(3, -2)$ about X-axis

$$P = \begin{bmatrix} 3 & -2 & 1 \end{bmatrix}^T, \quad R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_x \cdot P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$\Rightarrow P'(3, 2)$$

(ii) Reflect $P(-4, 5)$ about Y-axis

$$P = \begin{bmatrix} -4 & 5 & 1 \end{bmatrix}^T, \quad R_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_y \cdot P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}$$

$$\Rightarrow P'(4, 5)$$

(iii) Reflect $P(2, -3)$ about line $y = x$

$$P = \begin{bmatrix} 2 & -3 & 1 \end{bmatrix}^T, \quad R_{y=x} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_{y=x} \cdot P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 1 \end{bmatrix}$$

$$\Rightarrow P'(-3, 2)$$

9. Properties

- Reflection preserves the size and shape of the object.
- It is a rigid body transformation.
- Distance between any two points remains unchanged.
- $R^{-1} = R$ (Reflection matrix is its own inverse)
- Determinant of $R = -1$

10. Applications

- Creating mirror effects in computer graphics.
- Used in symmetry drawing and patterns.
- Image processing (horizontal / vertical flip).
- Used in animation, CAD/CAM and games.

11. Advantages

- ✓ Very simple and easy to implement.
- ✓ Requires less computation.
- ✓ Useful for symmetry and mirroring operations.
- ✓ Preserves size and shape of the object.

12. Limitations

- ✗ Only changes orientation and position.
- ✗ Does not produce realistic 3D effects.
- ✗ Not sufficient for complex transformations.

13. Important Points

- Reflection is specified by a reference line (axis).
- Sign of coordinate(s) is changed according to axis.
- Determinant of reflection matrix = -1.
- Reflection is its own inverse transformation.

Typical Exam Questions

1. Define 2D reflection with example.
2. Find the matrix for reflection about X-axis and verify.
3. Reflect the point $P(2, -5)$ about Y-axis.
4. Write the properties and applications of 2D reflection.

Summary

2D reflection creates a mirror image of an object with respect to a line (axis) such as X-axis, Y-axis, origin, $y = x$ or $y = -x$. It is represented using 3×3 reflection matrices in homogeneous coordinates. Reflection preserves size and shape but changes orientation. It is widely used in graphics, animation and image processing.

3D Translation

1. Introduction

Translation is a basic transformation in which every point of an object is shifted (moved) from its original position by a fixed distance along the X, Y and Z axes.

2. Definition

If a point P(x, y, z) is translated by t_x units along X-axis, t_y units along Y-axis and t_z units along Z-axis, the new position P′ (x′, y′, z′) is given by

$$\begin{aligned} x' &= x + t_x \quad , \\ y' &= y + t_y \quad , \\ z' &= z + t_z \quad , \end{aligned}$$

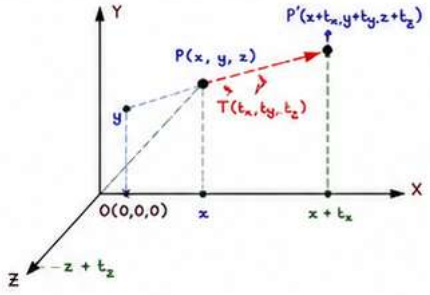
3. Translation Vector

The translation can be represented by a translation vector T.

$$T = (t_x \ , \ t_y, \ t_z)$$

New position = Old position + T

4. Geometrical Representation



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point P(x, y, z) is written as

$$P = [x \quad y \quad z \quad 1]^T$$

The matrix form of 3D translation is

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The new position is

$$P' = T(t_x, t_y, t_z) \cdot P$$

6. Derivation of Translation Matrix

Given,

$$x' = x + t_x \ , \quad y' = y + t_y \ , \quad z' = z + t_z$$

In homogeneous form,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Therefore,

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and $P' = T \cdot P$

7. Example

Translate the point P(3, 2, 1) by $t_x = 4$, $t_y = -3$, $t_z = 2$.

Solution :

$$P = [3 \quad 2 \quad 1 \quad 1]^T$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = T \cdot P$$

$$= \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3+4 \\ 2-3 \\ 1+2 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \\ 3 \\ 1 \end{bmatrix}$$

So, P′ (7, −1, 3)

8. Translation in Different Directions

Direction	t_x	t_y	t_z	Effect
Along +X axis	+a	0	0	Moves object +a units along X
Along −X axis	−a	0	0	Moves object −a units along X
Along +Y axis	0	+b	0	Moves object +b units along Y
Along −Y axis	0	−b	0	Moves object −b units along Y
Along +Z axis	0	0	+c	Moves object +c units along Z
Along −Z axis	0	0	−c	Moves object −c units along Z

9. Properties

- Translation does not change the shape or size of the object.
- Only the position of the object changes.
- Parallel lines remain parallel after translation.
- It is a rigid body transformation.
- The determinant of translation matrix = 1.

10. Applications

- ✓ Used in moving objects in 3D scenes.
- ✓ Used in animation and games.
- ✓ Used in CAD/CAM and robot motion.
- ✓ Used in scientific visualization and simulations.

11. Advantages

- ✓ Very simple and easy to implement.
- ✓ Requires less computation.
- ✓ Preserves shape, size and orientation.
- ✓ Useful for real-time graphics applications.

12. Limitations

- ✗ Translation alone cannot rotate or scale object.
- ✗ Not useful for perspective changes.
- ✗ Does not affect the orientation of the object.

13. Important Points

- Formula : $x' = x + t_x$, $y' = y + t_y$, $z' = z + t_z$.
- Matrix : $T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Translation vector : $T = (t_x \ , \ t_y, \ t_z)$
- In homogeneous coordinates, $P' = T \cdot P$

Typical Exam Questions

1. Define 3D translation.
2. Derive the translation matrix for 3D translation.
3. Translate the point (2, −1, 3) by (5, 2, −4).
4. Write properties and applications of 3D translation.
5. Verify that $P' = T \cdot P$.

Summary

3D translation moves every point of an object by fixed distances t_x, t_y and t_z along the X, Y and Z axes respectively. It is represented by the translation vector $T(t_x, t_y, t_z)$ and the 4×4 translation matrix. It preserves the shape, size and orientation of the object and is widely used in computer graphics.

3D Rotation

1. Introduction

Rotation is a transformation in which an object is rotated about one of the coordinate axes (X, Y or Z) by a specified angle θ (counter-clockwise positive) keeping the axis fixed.

2. Definition

If a point $P(x, y, z)$ is rotated about one of the coordinate axes by an angle θ, the new position $P'(x', y', z')$ is obtained by rotation transformation.

3. Rotation Vector

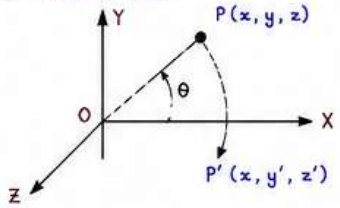
Rotation can also be represented by a rotation vector (angle of rotation).

Positive θ → Counter-clockwise rotation

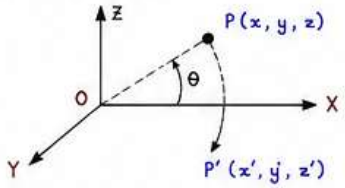
Negative θ → Clockwise rotation

4. Geometrical Representation

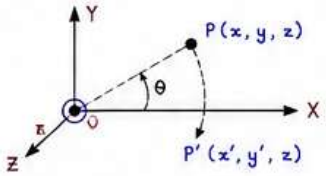
(i) About X-axis



(ii) About Y-axis



(iii) About Z-axis



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point $P(x, y, z)$ is written as

$$P = [x \quad y \quad z \quad 1]^T$$

The rotation transformation is represented by a 4×4 matrix R.

$$P' = R \cdot P$$

6. Rotation Matrices

(i) Rotation about X-axis by angle θ

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

In matrix form,

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_x(\theta) \cdot P$$

(ii) Rotation about Y-axis by angle θ

$$x' = x\cos\theta + z\sin\theta$$
$$y' = y$$
$$z' = -x\sin\theta + z\cos\theta$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_y(\theta) \cdot P$$

(iii) Rotation about Z-axis by angle θ

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_z(\theta) \cdot P$$

7. General Form

If R is the rotation matrix (R_x, R_y or R_z), then the new position is

$$P' = R \cdot P$$

where P and P' are position vectors in homogeneous coordinates.

8. Example

Rotate the point $P(1, 2, 3)$ by 90° about Z-axis (counter-clockwise).

Here, $\theta = 90° \Rightarrow \cos 90° = 0, \sin 90° = 1$

$$R_z(90°) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P = [1 \quad 2 \quad 3 \quad 1]^T$$

$$P' = R_z(90°) \cdot P = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 3 \\ 1 \end{bmatrix}$$

So, $P' = (-2, 1, 3)$

9. Special Cases (Common Angles)

Axis	θ	cos θ	sin θ
X	0°	1	0
	90°	0	1
	180°	−1	0
	270°	0	−1
Y	0°	1	0
	90°	0	1
	180°	−1	0
	270°	0	−1
Z	0°	1	0
	90°	0	1
	180°	−1	0
	270°	0	−1

10. Properties

- Rotation preserves the size and shape of the object.
- It only changes the orientation and position of the object.
- Determinant of rotation matrix = 1.
- $R^{-1}(\theta) = R(-\theta)^T = R(-\theta)$.
- Rotation matrices are orthogonal.

11. Applications

- Used in CAD/CAM and 3D modeling.
- Used in animation, games and VR.
- Used in robotics for orientation control.
- Used in scientific visualization.

12. Advantages

- ✓ Easy to implement using matrices.
- ✓ Efficient for complex 3D transformations.
- ✓ Preserves distances and proportions.
- ✓ Useful in real-time graphics.

12. Limitations

- × Rotation about an arbitrary axis requires additional transformations.
- × Accumulated errors in multiple rotations may cause distortion.
- × Not suitable for scaling or translation.

14. Important Points

- Rotation is about one of the principal axes.
- Always use homogeneous coordinates.
- Positive θ → Counter-clockwise rotation (right-hand rule).
- Rotation matrix is 4×4 and orthogonal.
- $P' = R_x(\theta) \cdot P$ or $R_y(\theta) \cdot P$ or $R_z(\theta) \cdot P$

Typical Exam Questions

1. Define 3D rotation with example.
2. Derive the rotation matrix about X-axis.
3. Derive the rotation matrix about Y-axis.
4. Derive the rotation matrix about Z-axis.
5. Rotate the point (2, 1, 3) by 90° about Y-axis.
6. Write properties and applications of 3D rotation.

Summary

3D rotation rotates an object about X, Y or Z axis by an angle θ. It is represented by 4×4 rotation matrices $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$. The transformation is rigid and preserves size, shape and distances. It is widely used in 3D graphics, animation, robotics and scientific visualization.

Unit - III

3D Scaling

1. Introduction

Scaling (or size transformation) is a basic transformation in which the size of an object is increased or decreased with respect to a fixed reference point (usually the origin) by scale factors along X, Y and Z axes.

2. Definition

If a point $P(x, y, z)$ is scaled by factors S_x along X-axis, S_y along Y-axis and S_z along Z-axis, the new position $P'(x', y', z')$ is given by

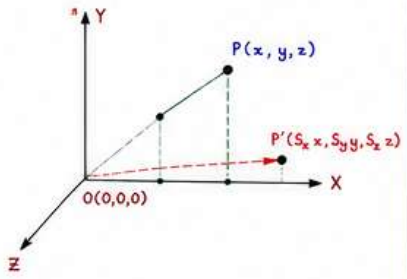
$$x' = S_x \cdot x$$
$$y' = S_y \cdot y$$
$$z' = S_z \cdot z$$

3. Scale Factors

* $S_x > 1$ → Enlargement along X-axis
* $0 < S_x < 1$ → Reduction along X-axis
* $S_x = 1$ → No change along X-axis

(Similary for S_y and S_z)

4. Geometrical Representation



5. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, a point $P(x, y, z)$ is written as

$$P = [x \quad y \quad z \quad 1]^T$$

The scaling transformation is represented by a 4×4 scaling matrix S.

$$S(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The new position is

$$P' = S \cdot P$$

6. Derivation of Scaling Matrix

Given $x' = S_x x$, $y' = S_y y$, $z' = S_z z$

In homogeneous form,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Therefore, $S(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

and

$$P' = S(S_x, S_y, S_z) \cdot P$$

7. Examples

(i) Scaling by $S_x = 2$, $S_y = 3$, $S_z = 4$

Matrix,

$$S = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let $P(1, 2, 3)$

$$P = [1 \quad 2 \quad 3 \quad 1]^T$$

$$P' = S \cdot P = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 6 \\ 12 \\ 1 \end{bmatrix} \Rightarrow P'(2, 6, 12)$$

(ii) Scaling by $S_x = \frac{1}{2}$, $S_y = \frac{1}{2}$, $S_z = \frac{1}{2}$ (Reduction)

$$S = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let $P(4, -2, 6)$

$$P = [4 \quad -2 \quad 6 \quad 1]^T$$

$$P' = S \cdot P = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ -2 \\ 6 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ -1 \\ 3 \\ 1 \end{bmatrix}$$

$$\Rightarrow P'(2, -1, 3)$$

8. Types of Scaling

Type of Scaling	Scale Factors (S_x, S_y, S_z)	Effect
Uniform Scaling	$S_x = S_y = S_z = k$	Same scale in all directions
Non-uniform Scaling	$S_x \neq S_y \neq S_z$	Different scale along axes
Enlargement	$S_x > 1$, $S_y > 1$, $S_z > 1$	Object size increases
Reduction	$0 < S_x < 1$, $0 < S_y < 1$, $0 < S_z < 1$	Object size decreases
Reflection by Scaling	One scale factor is negative	Object is flipped along that axis

9. Properties

* Scaling does not change the shape, only size.
* Only the distance from the reference point (origin) changes.
* Parallel lines remain parallel after scaling.
* Determinant of scaling matrix

$$|S| = S_x \cdot S_y \cdot S_z$$

* Inverse of scaling matrix

$$S^{-1} = S\left(\frac{1}{S_x}, \frac{1}{S_y}, \frac{1}{S_z}\right)$$

10. Applications

* Used in zooming in/out of 3D objects.
* Used in animation and simulations.
* Used in computer-aided design (CAD).
* Used in gaming and virtual reality.

11. Advantages

✓ Simple and easy to implement.
✓ Requires less computation.
✓ Preserves shape and orientation.
✓ Useful in real-time graphics applications.

12. Limitations

✗ Scaling about a point other than origin needs additional translation.
✗ Negative scaling results in reflection, which may not be desired.
✗ Excessive scaling may cause overflow or precision errors.

13. Important Points

* Formula : $x' = S_x x$, $y' = S_y y$, $z' = S_z z$
* Matrix : $S(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
* New position : $P' = S \cdot P$
* If $S_x = S_y = S_z = 1$ ⇒ No change.
* Origin remains fixed after scaling.

Typical Exam Questions

1. Define 3D scaling.
2. Derive the scaling matrix for 3D scaling.
3. Explain different types of 3D scaling.
4. Scale the point $(2, -1, 3)$ by $(2, 3, 4)$.
5. Write properties and applications of 3D scaling.

Summary

3D scaling changes the size of an object with respect to a fixed reference point (usually origin) using scale factors S_x, S_y and S_z along X, Y and Z axes.

The transformation is represented by a 4×4 scaling matrix.

It preserves the shape and orientation of the object and is widely used in graphics, animation, CAD and simulations.

3D Reflection

1. Introduction

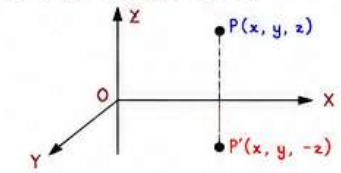
Reflection is a transformation that produces a mirror image of an object with respect to a reference plane (XY, YZ or ZX) or a coordinate plane (X = 0, Y = 0 or Z = 0).

2. Definition

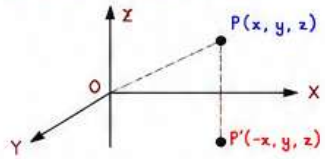
If a point P(x, y, z) is reflected with respect to a plane or coordinate plane, the new position P'(x', y', z') is obtained by changing the sign of the coordinate(s) perpendicular to that plane.

3. Geometrical Representation

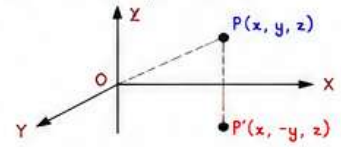
(i) About XY-plane (Z = 0)



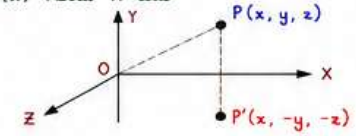
(ii) About YZ-plane (X = 0)



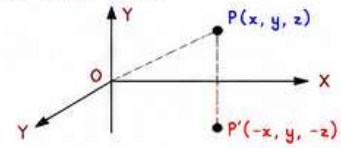
(iii) About ZX-plane (Y = 0)



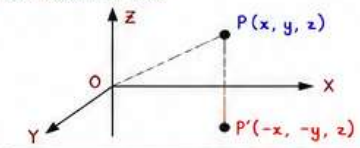
(iv) About X-axis



(v) About Y-axis



(vi) About Z-axis



4. Homogeneous Coordinates & Matrix Form

In homogeneous coordinates, P(x, y, z) is written as

$$P = [\,x \quad y \quad z \quad 1\,]^T$$

Reflection transformations are represented by 4 × 4 reflection matrices.

$$P' = R \cdot P$$

5. Reflection Matrices

Reflection about	Effect on (x, y, z)	Reflection Matrix R
XY-plane (Z = 0)	(x, y, z) → (x, y, −z)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
YZ-plane (X = 0)	(x, y, z) → (−x, y, z)	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
ZX-plane (Y = 0)	(x, y, z) → (x, −y, z)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
About X-axis	(x, y, z) → (x, −y, −z)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
About Y-axis	(x, y, z) → (−x, y, −z)	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
About Z-axis	(x, y, z) → (−x, −y, z)	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

6. General Form

If R is the reflection matrix, then

$$P' = R \cdot P$$

where P and P' are position vectors in homogeneous coordinates.

7. Examples

(i) Reflect P(3, 2, 4) about XY-plane (Z = 0).

$$P = [\,3 \quad 2 \quad 4 \quad 1\,]^T$$

$$R_{XY} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_{XY} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ -4 \\ 1 \end{bmatrix}$$

$$\Rightarrow P'(3, 2, -4)$$

(ii) Reflect P(1, −2, 3) about YZ-plane (X = 0).

$$P = [\,1 \quad -2 \quad 3 \quad 1\,]^T$$

$$R_{YZ} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_{YZ} \cdot P = \begin{bmatrix} -1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

$$\Rightarrow P'(-1, -2, 3)$$

(iii) Reflect P(−2, 3, −1) about Z-axis.

$$P = [\,-2 \quad 3 \quad -1 \quad 1\,]^T$$

$$R_Z = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = R_Z \cdot P = \begin{bmatrix} 2 & -3 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow P'(2, -3, -1)$$

8. Summary Table (Point Reflection)

Reflection about	New Coordinates (x', y', z')
XY-plane (Z = 0)	(x, y, −z)
YZ-plane (X = 0)	(−x, y, z)
ZX-plane (Y = 0)	(x, −y, z)
X-axis	(x, −y, −z)
Y-axis	(−x, y, −z)
Z-axis	(−x, −y, z)

9. Properties

- Reflection preserves the size and shape of the object.
- It changes only the orientation (handedness) of the object.
- It is its own inverse transformation. ($R^{-1} = R$)
- Determinant of reflection matrix = −1.
- Reflection matrices are orthogonal.

10. Applications

- ✓ Used in mirror images and symmetry operations.
- ✓ Used in computer graphics and animation.
- ✓ Used in CAD/CAM modeling.
- ✓ Used in games and virtual environments.

11. Advantages

- ✓ Simple to implement.
- ✓ Preserves dimensions and distances.
- ✓ Useful for symmetry and mirroring.
- ✓ Requires less computation.

12. Limitations

- ✗ Does not produce realistic 3D effects by itself.
- ✗ Only changes orientation, not position or size.
- ✗ Excessive reflections may cause confusion in complex scenes.

13. Important Points

- Reflection is a mirror transformation about a plane, axis or origin.
- One coordinate (or two) changes sign depending on the reference.
- Reflection matrix is orthogonal and determinant = −1.
- Origin remains fixed after reflection.

Typical Exam Questions

1. Define 3D reflection with example.
2. Derive reflection matrix about XY-plane.
3. Derive reflection matrix about YZ-plane.
4. Find the image of P(2, −1, 3) about ZX-plane.
5. Write properties and applications of 3D reflection.

Summary

3D reflection produces the mirror image of an object with respect to a plane, axis or origin. It is represented using 4 × 4 reflection matrices in homogeneous coordinates. Reflection preserves size and shape, changes only orientation, and is widely used in computer graphics, CAD/CAM, animation and simulations.

Inverse Transformation

1. Introduction

Inverse transformation is the operation which reverses the effect of a given transformation and restores the original object.

If T is a transformation and T^{-1} is its inverse, then:

$T^{-1}(T(P)) = P$ and $T(T^{-1}(P)) = P$

2. General Form

If the transformation matrix is T, then the inverse transformation is:

$$P = T \cdot P' \implies P' = T^{-1} \cdot P$$

where P : original point

P' : transformed point

3. Inverse of Homogeneous Transformation Matrix

For any non-singular matrix T,

$$T^{-1} = \frac{1}{|T|}\,\text{adj}(T)$$

where $|T|$ = determinant of T

$\text{adj}(T)$ = adjoint (transpose of cofactor)

4. 2D Basic Transformations and their Inverses

(i) Translation by (t_x, t_y)

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling by (S_x, S_y)

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(iii) Rotation by θ (about origin)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R^T$$

(Inverse rotation is rotation by $-\theta$)

5. 2D Reflection and their Inverses

Reflection about	Matrix R	Inverse R^{-1}
X-axis	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Y-axis	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Origin	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Line $y = x$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Line $y = -x$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

6. 3D Basic Transformations and their Inverses

(i) Translation by (t_x, t_y, t_z)

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling by (S_x, S_y, S_z)

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 1/S_x & 0 & 0 & 0 \\ 0 & 1/S_y & 0 & 0 \\ 0 & 0 & 1/S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(iii) Rotation about Principal Axes

About X-axis by θ

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x^{-1} = R_x^T = R_x(-\theta)$$

About Y-axis by θ

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y^{-1} = R_y^T = R_y(-\theta)$$

About Z-axis by θ

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_z^{-1} = R_z^T = R_z(-\theta)$$

7. 3D Reflections and their Inverses

Reflection about	Matrix R	Inverse R^{-1}
XY-plane ($Z = 0$)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
YZ-plane ($X = 0$)	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
ZX-plane ($Y = 0$)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
About Origin	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8. Composite Transformation

If a sequence of transformations is given by

$$T = T_n \cdot T_{n-1} \cdot \cdots \cdot T_2 \cdot T_1$$

then its inverse is

$$T^{-1} = T_1^{-1} \cdot T_2^{-1} \cdot \cdots \cdot T_{n-1}^{-1} \cdot T_n^{-1}$$

(i.e., Reverse the order and take inverse of each.)

9. Properties of Inverse Transformation

- $T\,T^{-1} = T^{-1}\,T = I$ (Identity matrix)
- Inverse of inverse is original: $(T^{-1})^{-1} = T$
- Inverse of a product is reverse product of inverses.
- For rotation matrices, $R^{-1} = R^T$ (orthogonal).
- Determinant of inverse: $|T^{-1}| = \frac{1}{|T|}$

10. Applications

- ✓ Restoring original object in modeling and animation.
- ✓ Reversing viewing transformations (camera).
- ✓ Used in hierarchical modeling (Scene graph).
- ✓ Essential in robotic arms and motion control.
- ✓ Used in computer vision and graphics pipelines.

Typical Exam Questions

1. Find the inverse of 2D translation matrix.
2. Find the inverse of 3D scaling matrix.
3. Show that inverse of rotation matrix $R(\theta)$ is $R(-\theta)$.
4. Find inverse of a composite transformation.
5. Write properties and applications of inverse transformation.

Summary

Inverse transformation undoes the effect of a transformation and restores the original object. For translation, we negate the translation factors; for scaling, we take reciprocal of scale factors; for rotation, we change the sign of the angle (or transpose the matrix); and for reflection, the inverse is the same matrix (since $R^2 = I$). These are widely used in graphics, animation, CAD/CAM, and simulations.

Important Points

- Translation inverse → negate (t_x, t_y, t_z)
- Scaling inverse → reciprocal of scale factors
- Rotation inverse → angle changes to $-\theta$
- Reflection inverse → same as reflection matrix
- Order is reversed in composite inverse

Inverse Transformation

1. Introduction

Inverse transformation is the operation which reverses the effect of a given transformation and restores the original object.

If T is a transformation and T^{-1} is its inverse, then:

$T^{-1}(T(P)) = P$ and $T(T^{-1}(P)) = P$

2. General Form

If the transformation matrix is T, then the inverse transformation is:

$$P = T \cdot P' \implies P' = T^{-1} \cdot P$$

where P : original point

P' : transformed point

3. Inverse of Homogeneous Transformation Matrix

For any non-singular matrix T,

$$T^{-1} = \frac{1}{|T|}\,\text{adj}(T)$$

where $|T|$ = determinant of T

$\text{adj}(T)$ = adjoint (transpose of cofactor)

4. 2D Basic Transformations and their Inverses

(i) Translation by (t_x, t_y)

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling by (S_x, S_y)

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(iii) Rotation by θ (about origin)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R^T$$

(Inverse rotation is rotation by $-\theta$)

5. 2D Reflection and their Inverses

Reflection about	Matrix R	Inverse R^{-1}
X-axis	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Y-axis	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Origin	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Line $y = x$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Line $y = -x$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

6. 3D Basic Transformations and their Inverses

(i) Translation by (t_x, t_y, t_z)

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling by (S_x, S_y, S_z)

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 1/S_x & 0 & 0 & 0 \\ 0 & 1/S_y & 0 & 0 \\ 0 & 0 & 1/S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(iii) Rotation about Principal Axes

About X-axis by θ

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x^{-1} = R_x^T = R_x(-\theta)$$

About Y-axis by θ

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y^{-1} = R_y^T = R_y(-\theta)$$

About Z-axis by θ

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_z^{-1} = R_z^T = R_z(-\theta)$$

7. 3D Reflections and their Inverses

Reflection about	Matrix R	Inverse R^{-1}
XY-plane ($Z = 0$)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
YZ-plane ($X = 0$)	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
ZX-plane ($Y = 0$)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
About Origin	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8. Composite Transformation

If a sequence of transformations is given by

$$T = T_n \cdot T_{n-1} \cdot \cdots \cdot T_2 \cdot T_1$$

then its inverse is

$$T^{-1} = T_1^{-1} \cdot T_2^{-1} \cdot \cdots \cdot T_{n-1}^{-1} \cdot T_n^{-1}$$

(i.e., Reverse the order and take inverse of each.)

9. Properties of Inverse Transformation

- $T\,T^{-1} = T^{-1}\,T = I$ (Identity matrix)
- Inverse of inverse is original: $(T^{-1})^{-1} = T$
- Inverse of a product is reverse product of inverses.
- For rotation matrices, $R^{-1} = R^T$ (orthogonal).
- Determinant of inverse: $|T^{-1}| = \frac{1}{|T|}$

10. Applications

- ✓ Restoring original object in modeling and animation.
- ✓ Reversing viewing transformations (camera).
- ✓ Used in hierarchical modeling (Scene graph).
- ✓ Essential in robotic arms and motion control.
- ✓ Used in computer vision and graphics pipelines.

Typical Exam Questions

1. Find the inverse of 2D translation matrix.
2. Find the inverse of 3D scaling matrix.
3. Show that inverse of rotation matrix $R(\theta)$ is $R(-\theta)$.
4. Find inverse of a composite transformation.
5. Write properties and applications of inverse transformation.

Summary

Inverse transformation undoes the effect of a transformation and restores the original object. For translation, we negate the translation factors; for scaling, we take reciprocal of scale factors; for rotation, we change the sign of the angle (or transpose the matrix); and for reflection, the inverse is the same matrix (since $R^2 = I$). These are widely used in graphics, animation, CAD/CAM, and simulations.

Important Points

- Translation inverse → negate (t_x, t_y, t_z)
- Scaling inverse → reciprocal of scale factors
- Rotation inverse → angle changes to $-\theta$
- Reflection inverse → same as reflection matrix
- Order is reversed in composite inverse

Composite Transformation

1. Introduction

Composite transformation is the combination of two or more basic transformations performed one after another. These transformations are applied in a sequence, and the overall effect is equivalent to a single transformation.

2. General Form

If a point P is transformed by T_1, then the result is further transformed by T_2, the composite transformation T is:

$$T = T_2 \cdot T_1$$

and

$$P' = T_r \cdot P = T_2 (T_1 \, P)$$

3. Order of Transformations

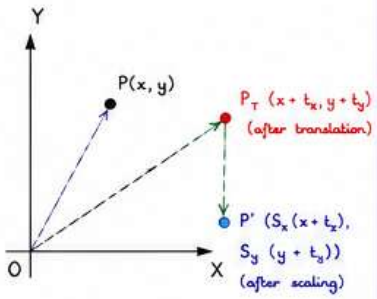
Matrix multiplication is not commutative.

$$T_2 \cdot T_1 \neq T_1 \cdot T_2$$

Therefore, the order in which transformations are applied is very important.

Geometrical Representation (2D)

Example: Translate by (t_x, t_y) then scale by (S_x, S_y)



Important Note

The transformation that is written on the right is applied first.

$$T = T_2 \cdot T_1$$

means: First apply T_2, then apply T_2.

4. 2D Composite Transformations

(i) Translation followed by Scaling

First translate by (t_x, t_y), then scale by (S_x, S_y)

$$T = S \cdot T_r$$

Where,

$$T_r = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So,

$$T = \begin{bmatrix} S_x & 0 & S_x \, t_x \\ 0 & S_y & S_y \, t_y \\ 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling followed by Translation

First scale by (S_x, S_y), then translate by (t_x, t_y)

$$T = T_r \cdot S$$

So,

$$T = \begin{bmatrix} S_x & 0 & t_x \\ 0 & S_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

5. Example (2D)

Rotate a point 90° about origin, then translate by (3, 2). Find the composite matrix and the new position of P(1, 2).

Rotation (90° about origin):

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad T_r = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite transformation:

$$T = T_r \cdot R = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, $P = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$, $P' = T_r \cdot P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

So, new position P'(1, 1)

6. 3D Composite Transformations

(i) Translation followed by Scaling

$$T = S \cdot T_r$$

$$T_r = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} S_x & 0 & 0 & S_x \, t_x \\ 0 & S_y & 0 & S_y \, t_y \\ 0 & 0 & S_z & S_z \, t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) Rotation about Z-axis (θ) followed by Translation (t_x, t_y, t_z)

$$T = T_r \cdot R_z(\theta)$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_r = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. Example (3D)

Rotate a point 90° about Z-axis, then translate by (3, 2, 5). Find the composite matrix and the new position of P(1, 0, 2).

$$R_z(90°) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_r = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = T_r \, R_z(90°) = \begin{bmatrix} 0 & -1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix}, \quad P' = TP = \begin{bmatrix} 3 \\ 3 \\ 7 \\ 1 \end{bmatrix}$$

So, new position P'(3, 3, 7)

8. Properties

- Composite transformation is the product of individual transformation matrices.
- Matrix multiplication is associative: $(T_3 \, T_2) \, T_1 = T_3 \, (T_2 \, T_1)$
- Not commutative in general: $T_2 \, T_1 \neq T_1 \, T_2$
- Order of transformations is important.

9. Applications

- ✓ Used to perform complex operations with a single matrix.
- ✓ Essential in hierarchical modeling.
- ✓ Used in animation and robotics.
- ✓ Used in camera/view transformations in 3D graphics.

10. Advantages

- ✓ Reduces multiple steps to a single matrix.
- ✓ Easy to implement using matrix operations.
- ✓ Saves computation time.
- ✓ Provides smooth and accurate results.

11. Limitations

- × Incorrect order gives wrong results.
- × Accumulation of floating-point errors in many operations.
- × Harder to interpret geometrically.

12. Important Points

- $T = T_n \, T_{n-1} \cdots T_2 \, T_1$
- Rightmost transformation is applied first.
- Use homogeneous coordinates.
- Inverse of composite transformation: $T^{-1} = (T_n \, T_{n-1} \cdots T_1)^{-1} = T_1^{-1} \, T_2^{-1} \cdots T_n^{-1}$
- For rotation matrices, inverse = transpose.

Typical Exam Questions

1. Define composite transformation with example.
2. Show that $T = T_2 \, T_1$ and $P' = T_2 \, (T_1 \, P)$.
3. Find the composite matrix for:
   (i) Translation then scaling (2D)
   (ii) Rotation then translation (3D)
4. Find the final position of a point after given composite transformations.
5. Prove that $(T_3 \, T_2 \, T_1)^{-1} = T_1^{-1} \, T_2^{-1} \, T_3^{-1}$.

Summary

Composite transformation is the combination of two or more basic transformations applied in sequence. The overall transformation matrix is obtained by multiplying the individual matrices in the order of application. It is widely used in computer graphics, animation, CAD/CAM, robotics and simulations to perform complex operations efficiently.

World Coordinate System

1. Introduction

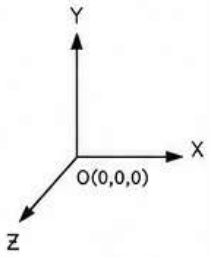
The world coordinate system (WCS) is a right-handed 3D coordinate system used to specify the position and orientation of objects in the world (real world or modeling environment). It is independent of any particular view or projection.

2. Definition

A world coordinate system is a fixed reference coordinate system in which all objects in a scene are defined. It provides a common framework for modeling, placing and transforming objects.

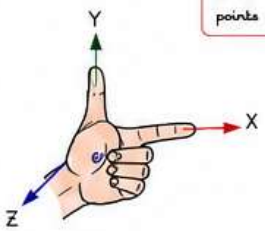
3. Axes Orientation

WCS is a right-handed orthogonal coordinate system.



Right-Hand Rule

If the thumb, index and middle finger of the right hand are mutually perpendicular, and the index finger points along +X, the middle finger points along +Y, then the thumb points along +Z.



4. Coordinates in WCS

Any point P in world space is represented by a triplet of coordinates.

$$P = (x_w, y_w, z_w)$$

where,

- x_w – distance along X-axis
- y_w – distance along Y-axis
- z_w – distance along Z-axis

5. Homogeneous Representation

In homogeneous coordinates, a point in WCS is written as:

$$P_w = \begin{bmatrix} x_w & z_w \\ y_w & z_w \\ z_w & 1 \end{bmatrix}^T$$

All geometric tranformations are performed using 4 × 4 matrices.

6. Features of WCS

- It is a right-handed Cartesian coordinate system.
- Origin (0,0,0) is fixed in the world.
- Axes are mutually perpendicular.
- It is independent of the viewing or projection parameters.
- All objects are specified with respect to WCS.
- It serves as the starting point for all transformations.

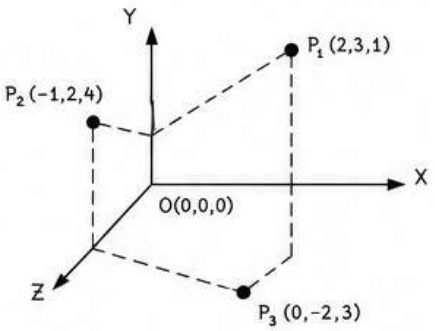
7. Example

Consider three points in world coordinate system.

P_1 (2, 3, 1)

P_2 (−1, 2, 4)

P_3 (0, −2, 3)



8. World Coordinate System VS Other Coordinate Systems

Coordinate System	Purpose / Use	Depends on
World Coordinate System (WCS)	Defines the position of objects in the real world / modeling environment.	Independent (Fixed in world)
View Coordinate System (VCS)	Positions the world with respect to the viewer (camera).	Viewing transformation
Device Coordinate System (DCS)	Represents coordinates on the actual device (screen).	Device (Screen)
Normalized Device Coordinate System (NDCS)	Intermediate system after projection, range (−1 to +1).	Projection transformation

9. Transformation Chain

Objects defined in WCS are transformed to finally appear on the screen.

WCS → VCS → NDCS → DCS

World Coordinate System — View Coordinate System — Normalized Device Coordinate System — Device Coordinate System

(Modeling) ← - - - - - - - - - - - - - - - - - - → (Display)

10. Importance of WCS

- Provides a common and consistent reference for all objects.
- Simplifies object modeling.
- Enables easy positioning and movement of objects.
- Foundation for further transformations (translation, rotation, scaling, etc.).
- Essential in animation, simulation, robotics, CAD/CAM, and 3D games.

11. Properties

- Right-handed system.
- Origin (0,0,0) is fixed.
- Axes are orthogonal.
- Unit length along each axis.
- Used to define all objects in the scene.

12. Advantages

- ✓ Provides a universal reference frame.
- ✓ Easy to understand and implement.
- ✓ Independent of camera/view.
- ✓ Useful in complex 3D modeling.
- ✓ Essential for all transformation operations.

13. Limitations

- ✗ Does not consider the viewer or display.
- ✗ Objects may appear differently from different viewpoints.
- ✗ Need additional transformations for viewing and display.

14. Applications

- 3D object modeling in CAD systems.
- Animation and movie production.
- Virtual reality and simulation.
- Robotics and automation.
- Scientific visualization.
- 3D games and interactive graphics.

15. Summary

The World Coordinate System is a right-handed 3D coordinate system that serves as the global reference for defining and positioning objects in a scene. It is independent of the viewer and display system and forms the basis for all geometric transformations in computer graphics.

Key Points

- WCS is the starting coordinate system.
- Right-handed: +X → index, +Y → middle, +Z → thumb.
- All objects are defined in WCS.
- Transformation chain: WCS → VCS → NDCS → DCS

Typical Exam Questions

1. Define world coordinate system.
2. Explain the orientation of world coordinate system.
3. Write the homogeneous representation of a point in WCS.
4. Distinguish between WCS, VCS, and DCS.
5. Draw the world coordinate system and explain.

Topic – 14 Screen Coordinate System

Screen Coordinate System

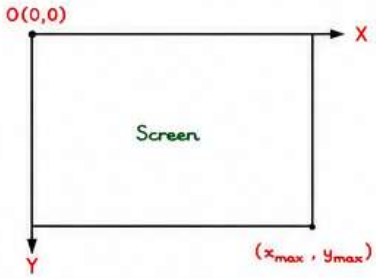
1. Introduction

The screen coordinate system (SCS) is a 2D coordinate system used to specify the position of points on the display device (monitor). It is the final coordinate system in the graphics pipeline where objects are actually drawn (displayed).

2. Definition

Screen coordinate system is a 2D rectangular coordinate system whose origin is at the top-left corner of the screen and where the X-axis is to the right and Y-axis is downwards.

3. Axes Orientation



Important :

- Origin (0,0) is at top-left.
- X increases to the right.
- Y increases downwards.
- This is a left-handed coordinate system.

4. Coordinates in SCS

Any point P on the screen is represented as :

$$P = (x_s , y_s)$$

where,

x_s = horizontal distance from left edge

y_s = vertical distance from top edge

$x_s \in [0, x_{max}]$

$y_s \in [0, y_{max}]$

$(x_{max} + 1) \times (y_{max} + 1)$ are the total number of addressable screen positions (pixels).

5. Pixels and Addressability

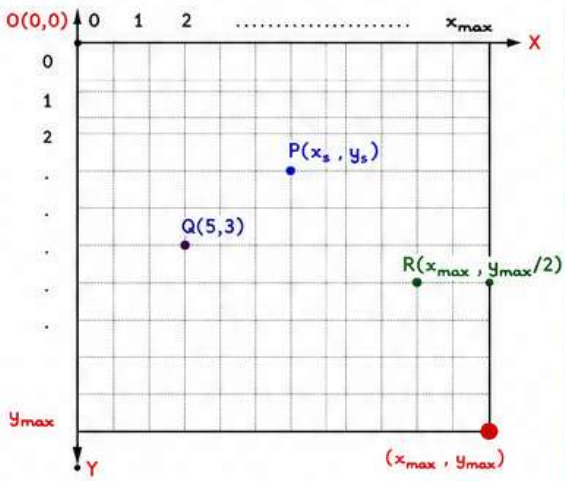
- Screen is made of pixels.
- Each pixel has integer coordinates.
- Smallest addressable unit is a pixel.
- Top-left pixel = (0, 0)
- Bottom-right pixel = (x_{max} , y_{max})

For an $N_x \times N_y$ screen :

$x_{max} = N_x - 1$

$y_{max} = N_y - 1$

6. Screen Coordinate System Diagram



7. Example

Assume a screen of size 800 × 600 pixels.

Then :

$x_s \in [0, 799]$

$y_s \in [0, 599]$

Top-left pixel = (0, 0)

Bottom-right pixel = (799, 599)

Point P(200, 150) is 200 pixels from left and 150 pixels from top.

8. World Coordinate System vs Screen Coordinate System

Feature	World Coordinate System	Screen Coordinate System
Type	3D Cartesian	2D Cartesian
Origin	Arbitrary (not fixed)	Top-left corner (0,0)
X-axis	Right	Right
Y-axis	Up	Down
Z-axis	Out of screen	Not present
Units	Real-world units	Pixels
Used for	Modeling objects	Displaying objects
Handedness	Right-handed	Left-handed

9. Transformation to SCS

To display an object on screen, coordinates are mapped from Normalized Device Coordinates (NDC) range (-1 to +1) to Screen Coordinates.

$$x_s = \frac{x_{ndc} + 1}{2} \times x_{max}$$

$$y_s = \frac{1 - y_{ndc}}{2} \times y_{max}$$

Note: y is inverted because SCS y increases downwards.

where $(x_{ndc} , y_{ndc}) \in [-1, 1]$

Example :

If $(x_{ndc} , y_{ndc}) = (0, 0)$ and screen = 800 × 600

Then, $x_s = 400$, $y_s = 300$ (center of screen)

10. Properties

- Fixed origin at top-left corner.
- X increases to the right.
- Y increases downwards.
- Integer coordinates.
- Left-handed 2D system.
- Used for mapping and display.
- Resolution dependent (depends on number of pixels).
- Final stage in graphics pipeline.

11. Advantages

- ✓ Easy to implement.
- ✓ Directly relates to display device.
- ✓ Simple integer arithmetic.
- ✓ Efficient for raster display.
- ✓ Compatible with pixel addressing.

12. Limitations

- ✗ Depends on screen resolution.
- ✗ Not suitable for object modeling.
- ✗ Y-axis direction is opposite of mathematical convention.
- ✗ Different devices → different pixel ranges.

13. Applications

- Displaying 2D and 3D graphics.
- Drawing lines, shapes, text.
- GUI (buttons, menus, windows).
- Animation and games.
- CAD, CAM and simulation displays.
- Image processing and visualization.

14. Key Points

- Screen coordinate system is 2D.
- Origin (0,0) is at top-left corner.
- X → right, Y → down.
- Coordinates are in pixels.
- Final coordinate system used for raster display.

15. Summary

Screen Coordinate System is a 2D left-handed coordinate system used to specify pixel positions on the display device. It has fixed origin at top-left, X-axis to the right and Y-axis downwards. It is the final coordinate system in the graphics pipeline where transformed objects are actually displayed on the screen.

Typical Exam Questions

1. Define screen coordinate system.
2. Explain axes orientation in SCS.
3. Write the range of coordinates in SCS.
4. Convert NDC (0.5, -0.5) to screen coordinates for a screen of size 1024 × 768.
5. Compare world coordinate system and screen coordinate system.

Key Formulas

1. Range: $0 \le x_s \le x_{max}$, $0 \le y_s \le y_{max}$
2. $x_{max} = N_x - 1$, $y_{max} = N_y - 1$
3. $x_s = (x_{ndc} + 1)\ 2 \times x_{max}$
4. $y_s = (1 - y_{ndc})\ 2 \times y_{max}$

15. Parallel Projection

1. Introduction

Parallel projection is a method of projecting 3D objects onto a projection plane using parallel lines of projection.

It is an approximation of perspective projection and is widely used in engineering drawings and computer graphics.

2. Definitions

- Projection Plane (P): The plane on which the object is projected.
- Projectors: The parallel lines from points on the object to the projection plane.
- Parallel Projection: Projection obtained using parallel projectors.

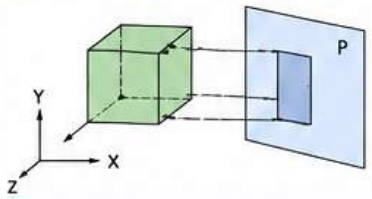
3. Types of Parallel Projection

Two common types:

- Orthographic Projection – Projectors are perpendicular to the projection plane.
- Oblique Projection – Projectors are inclined at an angle (α) to the projection plane.

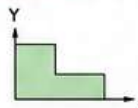
4. Orthographic Projection

In orthographic projection, projectors are perpendicular to the projection plane.

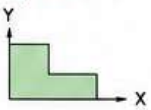


Orthographic Views

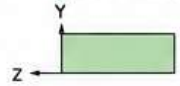
Front View (FV)



Top View (TV)

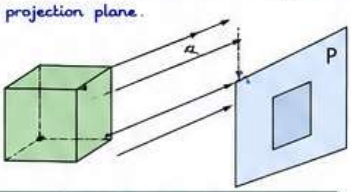


Right Side View (RSV)



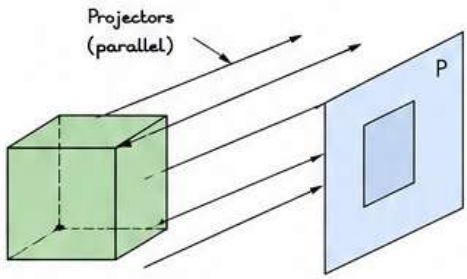
5. Oblique Projection

In oblique projection, projectors are parallel to each other and inclined at an angle α to the projection plane.



Common Values of α	
α (Angle)	Type
$\alpha = 0°$	Orthographic Projection
$0° < \alpha < 90°$	Oblique Projection
$\alpha = 90°$	Orthographic Projection

6. Diagram of Parallel Projection



- All projectors are parallel to each other.
- Object is projected onto plane P.

7. Mathematical Formulation

Let a point P have coordinates (x, y, z). It is projected onto the projection plane using parallel projectors.

(i) Orthographic Projection (along Z-axis)

$$P' = (x', y', 0) = (x, y, 0)$$

(ii) Oblique Projection (along direction $d = (l, m, n)$)

Let the projectors be inclined at an angle α.

Then,

$$x' = x + tl, \quad y' = y + tm, \quad z' = 0$$

where $t = -\frac{z}{n}$

Hence,

$$x' = x - \frac{z}{n} l, \quad y' = y - \frac{z}{n} m, \quad z' = 0$$

8. Comparison between Orthographic and Oblique

Feature	Orthographic Projection	Oblique Projection
Projectors	Perpendicular to plane	Inclined at angle α to plane
Angle α	$90°$	$0° < \alpha < 90°$
Size	True size in at least one view	General foreshortening
Depth Perception	No depth effect	Some depth effect
Applications	Engineering drawings, CAD	Pictorial views, illustrations

9. Properties

- Projectors are parallel.
- Parallel projection preserves parallelism.
- Orthographic projection preserves true size in at least one view.
- Oblique projection shows foreshortening.
- Simple and easy to implement.
- Does not show realistic depth (no perspective).

10. Applications

- ✓ Technical and engineering drawings.
- ✓ Architectural drawings and floor plans.
- ✓ CAD modeling and drafting.
- ✓ Circuit diagrams and schematics.
- ✓ Map projections and survey works.
- ✓ 2D representation of 3D objects in computer graphics.

11. Examples

Let a point P(2, 3, 5).

(i) Orthographic projection (along Z-axis):

$P' = (2, 3, 0)$

(ii) Oblique projection along $d = (1, 1, 1)$

$$t = -\frac{z}{n} = \frac{-5}{1} = -5$$

$$x' = 2 - 5(1) = -3$$

$$y' = 3 - 5(1) = -2$$

$$z' = 0$$

So, $P' = (-3, -2, 0)$

12. Advantages

- ✓ Simple and economical.
- ✓ Easy to construct.
- ✓ Useful for engineering and technical drawings.
- ✓ Preserves parallelism.
- ✓ Orthographic projection shows true size in one view.

13. Limitations

- ✗ Does not provide realistic depth perception.
- ✗ Orthographic projection gives no 3D effect.
- ✗ Oblique projection causes distortion in size.
- ✗ Not suitable for photorealistic visualization.
- ✗ Angles and lengths are not preserved in general (oblique).

14. Key Points

- ▸ Parallel projection uses parallel projectors.
- ▸ Two types: Orthographic ($\alpha = 90°$) and Oblique ($0° < \alpha < 90°$).
- ▸ Orthographic projectors are perpendicular to the plane.
- ▸ Oblique projectors are inclined at angle α.
- ▸ Parallel projection preserves parallelism.
- ▸ Used widely in engineering drawings and CAD.

15. Summary

Parallel projection is a simple projection technique using parallel projectors. It produces pictorial or orthographic views of 3D objects on a projection plane. Orthographic projection gives true size in at least one view, while oblique projection shows some foreshortening. It is widely used in technical fields where simplicity and accuracy are important.

Typical Exam Questions

1. Define parallel projection.
2. Differentiate between orthographic and oblique projection.
3. Derive the mathematical formulation for oblique projection.
4. Draw front, top and right side views of a given object.
5. List applications of parallel projection.

Important Formulas

- Orthographic projection (along Z-axis):
  $x' = x, \quad y' = y, \quad z' = 0$
- Oblique projection along $d = (l, m, n)$:
  $x' = x - \frac{z}{n} l, \quad y' = y - \frac{z}{n} m, \quad z' = 0$
  $t = -\frac{z}{n}$

Perspective Projection

1. Introduction

Perspective projection is a method of projecting a 3D object onto a 2D projection plane using projectors that converge at a single point called the Center of Projection (COP). It gives a realistic view similar to human vision.

Objects farther from the viewer appear smaller, and parallel lines appear to meet at the COP (vanishing point).

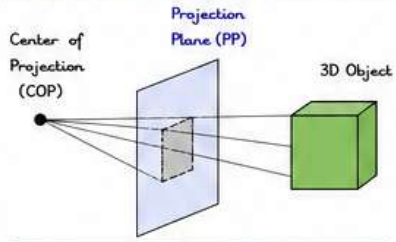
2. Definition

Perspective projection is a projection technique in which the projectors converge at a single point (COP) before intersecting the projection plane.

3. Key Terms

- Center of Projection (COP): The point from where projectors originate.
- Projection Plane (PP): The plane on which the 3D object is projected.
- Station Point (SP): The position of the viewer's eye.
- Picture Plane (PP): Same as projection plane.

4. Diagram of Perspective Projection



All projectors pass through COP and intersect the projection plane.

5. Types of Perspective Projection

(i) One-Point Perspective

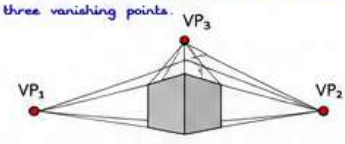
Object faces the viewer directly.
One set of parallel lines recede to one vanishing point.

(ii) Two-Point Perspective

Object is rotated so that two faces are visible. Two sets of parallel lines recede to two vanishing points.

(iii) Three-Point Perspective

Object is tilted so that three faces are visible. Three sets of parallel lines recede to three vanishing points.



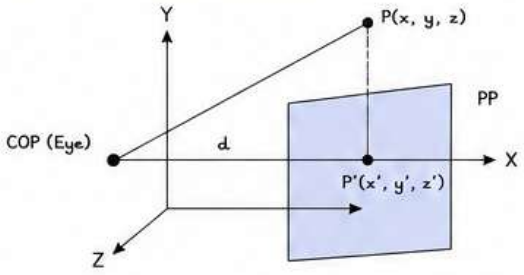
Key Points

- Perspective projection gives realistic view.
- Distant objects appear smaller.
- Parallel lines seem to meet at vanishing points.
- Number of vanishing points depends on orientation.
- Used widely in realistic graphics and visualization.

Typical Exam Questions

1. Define perspection projection.
2. Draw the diagram of perspective projection and explain.
4. Explain one-point, two-point and three-point perspective.
5. Compare parallel and perspective projection.

6. Perspective Projection Setup



Where,

- COP is taken as origin (0, 0, 0).
- Projection plane is placed at a distance d from COP along the Z-axis.
- d = Distance from COP to PP.

7. Mathematical Formulation

Let a 3D point P(x, y, z) be projected onto PP placed at Z = -d (d > 0).

Then, the perspective projection is given by:

$$x' = -d\,\frac{x}{z}\quad,\quad y' = -d\,\frac{y}{z}\quad,\quad z' = -d$$

Where:

- (x, y, z) = coordinates of the point in 3D space
- (x', y', z') = coordinates of the projected point on PP
- d = distance from COP to projection plane

8. Properties

- Projectors are concurrent at COP.
- Parallel lines (not parallel to PP) meet at vanishing points.
- Objects farther from the viewer appear smaller.
- Preserves the depth perception.
- Simple but computationally costlier than parallel projection.

9. Comparison: Parallel vs Perspective Projection

Feature	Parallel Projection	Perspective Projection
Projectors	Parallel to each other	Converge at COP
Size	Same	Smaller with distance
Depth perception	No	Yes (realistic)
Vanishing points	None	Present
Used in	Engineering drawings, CAD	Realistic views, games, animation, films

10. Example

Projection plane at Z = -d. Point P(4, 3, -8), d = 6.

Using formulas:

$$x' = -6 \times \frac{4}{-8} = 3$$

$$y' = -6 \times \frac{4}{-8} = 2.25$$

$$z' = -6$$

So, P' (3, 2.25, -6)

11. Applications

- Realistic 3D modeling in computer graphics.
- Animation and movie production.
- Video games and virtual reality.
- Architectural visualization.
- Simulation and training systems.

Key Formulas

1. Range: $0 \le x_s \le x_{max}$, $0 \le y_s \le y_{max}$
2. $x_{max} = N_x - 1$, $y_{max} = N_y - 1$
3. $x_s = (x_{ndc} + 1)\ 2 \times x_{max}$
4. $y_s = (1 - y_{mdc})\ 2 \times y_{max}$

12. Advantages

- ✓ Provides realistic view similar to human vision.
- ✓ Suitable for visualization and presentation.
- ✓ Distant objects appear smaller, adding depth perception.
- ✓ Used widely in graphics, animation and simulations.

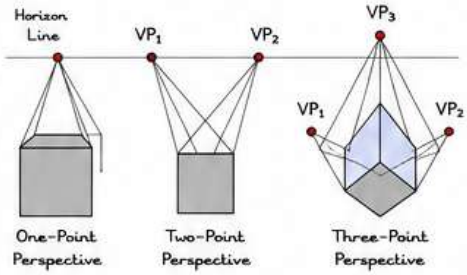
13. Limitations

- ✗ Computationally expensive.
- ✗ Objects completely behind the projection plane cannot be projected.
- ✗ Distortion in size and shape (not uniform).
- ✗ More complex than parallel projection.

14. Vanishing Points

Vanishing points are the points on the projection plane where parallel lines in space appear to converge.

- One-Point Perspective → 1 vanishing point.
- Two-Point Perspective → 2 vanishing points.
- Three-Point Perspective → 3 vanishing points.



15. Avplications

- Displaying 2D and 3D graphics.
- Drawing lines, shapes, text.
- GUI (buttons, menus, windowa).
- Animation and games.
- CAD, CAM and simulation displays.
- Image processing and visualization.

16. Key Points

- Perspective projection is a realistic projectors.
- Projectors converge at COP.
- Origin (0,0,0) is at the COP.
- Projection plane is at Z = -d.
- Objects farther from viewer appear smaller.
- Final coordinate system is used for display.

17. Summary

Perspective Projection uses converging projectors from a single point (COP) to project 3D objects onto a 2D plane. It produces realistic views with depth perception, where parallel lines meet at vanishing points. It is widely used in realistic graphics, animations, simulations and virtual environments.

Representation of 3D Object on 2D Screen

1. Introduction

Since computer displays are 2D devices, 3D objects must be represented on a 2D screen. This is achieved through a series of steps including modeling, coordinate transformation, projection, clipping and viewport mapping.

2. Object Representation

A 3D object is represented using geometric primitives such as:

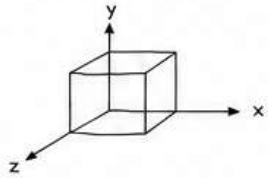
- Points
- Lines
- Polygons (Triangles, Quads, etc.)
- Surfaces
- Solids (Wireframe / Surface models)

3. General Process

The pipeline for representing a 3D object on 2D screen is:

4. 3D Object Example

Cube in 3D World



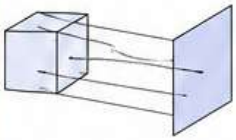
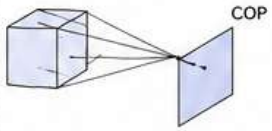
5. Coordinate Spaces Involved

Coordinate Space	Description
World (WC)	Object defined in real world coordinates.
View (VC)	Coordinates w.r.t. the camera (viewer).
Projection (PC)	After projection transformation.
Normalized Device Coordinates (NDC)	Standard cube with range -1 to +1.
Screen (SC)	Actual 2D screen coordinates (pixels).

6. Projection Transformation

Converts 3D coordinates to 2D (actually 3D to 3D but with depth information) using:

- Parallel Projection
- Perspective Projection

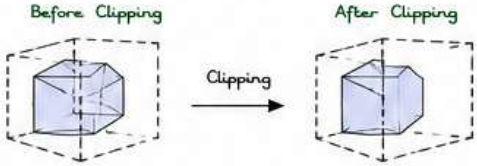
Parallel Projection	Perspective Projection
	
Projectors are parallel. Objects appear similar in size.	Projectors converge at COP. Objects farther away appear smaller.

7. Clipping

Removes the parts of objects that lie outside the viewing volume. Common clipping volumes:

- View Window (2D)
- View Volume (3D)

Clipping ensures only visible parts are processed.



8. Viewport Transformation

Maps the NDC (-1 to +1) coordinates to the actual screen coordinates (pixels).

It performs scaling and translation.

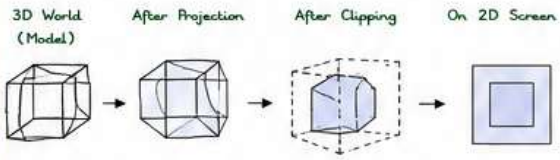
$$x_s = \frac{(x_{ndc} + 1)}{2} \times (x_{vmax} - x_{vmin}) + x_{vmin}$$

$$y_s = \frac{(y_{ndc} + 1)}{2} \times (y_{vmax} - y_{vmin}) + y_{vmin}$$

where (x_{vmin}, y_{vmin}) = lower-left corner of viewport

(x_{vmax}, y_{vmax}) = upper-right corner of viewport

9. Example: Cube Representation Pipeline



10. Common Display Modes

- Wireframe Model : Only edges are displayed.
- Surface Model : Only visible surfaces/polygons are displayed.
- Shaded Model : Surfaces are filled with colors (flat, Gouraud, Phong shading).
- Textured Model : Surfaces are mapped with images/textures.

11. Importance

- Allows 3D objects to be displayed on 2D devices.
- Foundation for computer graphics, animation, CAD, games, VR, simulators.
- Helps in visualizing complex objects in a human understandable form.

12. Advantages

- ✓ Realistic visualization of 3D scenes.
- ✓ Supports various applications.
- ✓ Efficient use of 2D display devices.
- ✓ Allows different projection techniques as per requirement.

13. Limitations

- ✗ Loss of depth information (2D display).
- ✗ Projection introduces distortion.
- ✗ Hidden surface problem may occur.
- ✗ Quality depends on projection and clipping methods.

14. Applications

- Computer games and animation.
- Virtual reality and simulations.
- CAD/CAM systems.
- Scientific visualization.
- Medical imaging.
- Training and flight simulators.
- GIS and mapping systems.

15. Key Points

- 3D objects must be projected to 2D screen for display.
- Projection can be parallel or perspective.
- Clipping removes invisible parts.
- Viewport transformation maps to actual screen pixels.
- Final output is the 2D image displayed on monitor.

16. Summary

A 3D object is represented on a 2D screen by transforming it from world coordinates to screen coordinates through a pipeline of transformations: modeling → world → view → projection → clipping → viewport mapping. This process ensures that the object appears in correct position, size and orientation on the display device.

Typical Exam Questions

1. Explain the process of representing a 3D object on 2D screen.
2. Draw the pipeline for 3D to 2D screen representation and explain each step.
3. Differentiate between parallel and perspective projection.
4. What is viewport transformation? Derive the equations.
5. Explain clipping and its importance in 3D graphics.

Key Formulas

- Viewport Transformation:

$$x_s = \frac{(x_{ndc} + 1)}{2} \times (x_{vmax} - x_{vmin}) + x_{vmin}$$

$$y_s = \frac{(y_{ndc} + 1)}{2} \times (y_{vmax} - y_{vmin}) + y_{vmin}$$

where NDC range is [-1, +1]

- Perspective Projection (General Form):

$$x' = -d \times \frac{x}{z}, \quad y' = -d \times \frac{y}{z}, \quad z' = -d$$

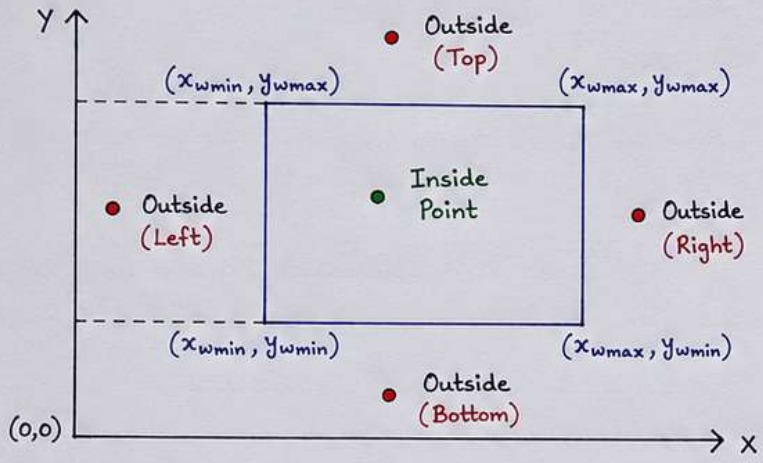
Topic-18 : POINT CLIPPING

RGPV IT 6th Sem

1. DEFINITION :

Point clipping is the process of determining whether a given point lies inside or outside a specified clipping window (rectangular window). If the point is inside the clipping window, it is accepted (visible); otherwise, it is rejected (invisible).

2. CLIPPING WINDOW :



3. INSIDE TEST (USING INEQUALITIES) :

A point P(x,y) is inside the clipping window if

$$x_{wmin} \leq x \leq x_{wmax}$$
$$y_{wmin} \leq y \leq y_{wmax}$$

If the above conditions are satisfied, the point is visible (inside). Otherwise, it is invisible (outside).

4. REGION CODE METHOD (4-BIT CODE) :

Each point can be assigned a 4-bit region code to quickly test its position with respect to the clipping window.

Bit	Position	Meaning
b_3 (8's)	Top	1 if $y > y_{wmax}$ else 0
b_2 (4's)	Bottom	1 if $y < y_{wmin}$ else 0
b_1 (2's)	Right	1 if $x > x_{wmax}$ else 0
b_0 (1's)	Left	1 if $x < x_{wmin}$ else 0

5. REGION CODE FORMATION :

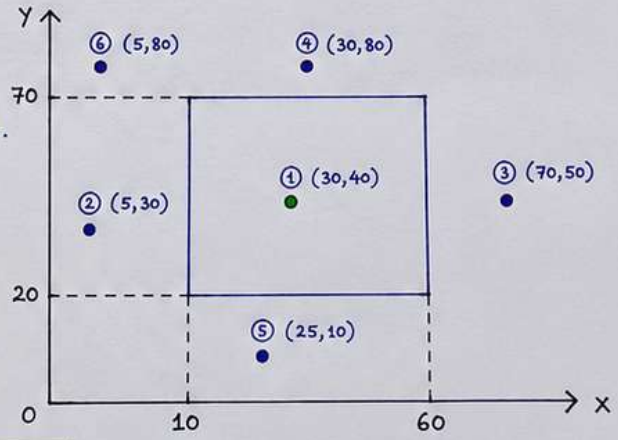
- Initialize code = 0000
- Set corresponding bit(s) to 1 based on the above conditions.
- If code = 0000 → point is inside (accept)
- If code ≠ 0000 → point is outside (reject)

6. EXAMPLE :

Clipping window limits : $x_{wmin} = 10$, $x_{wmax} = 60$
$y_{wmin} = 20$, $y_{wmax} = 70$

Find whether the following points are inside or outside using region code.

Point P(x,y)	Location	Region Code ($b_3 b_2 b_1 b_0$)	Inside/Outside
① P_1 (30, 40)	Inside	0 0 0 0	Inside (Accept)
② P_2 (5, 30)	Left	0 0 0 1	Outside (Reject)
③ P_3 (70, 50)	Right	0 0 1 0	Outside (Reject)
④ P_4 (30, 80)	Top	1 0 0 0	Outside (Reject)
⑤ P_5 (25, 10)	Bottom	0 1 0 0	Outside (Reject)
⑥ P_6 (5, 80)	Top-Left	1 0 0 1	Outside (Reject)



Note : Point clipping is a very simple and fast operation since it only checks the position of a single point with respect to the clipping window.

Line Clipping Algorithms

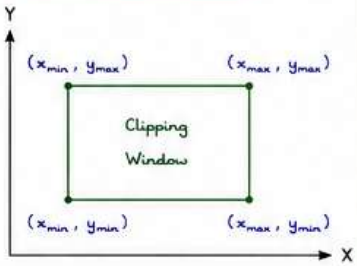
1. Introduction

Line clipping is the process of cutting off the portion of a line which lies outside the clipping window. Only the visible portion inside the window is retained for display. It improves display efficiency and removes invisible parts.

2. Clipping Window

A rectangular clipping window is defined by its boundaries :

x_{min}	left boundary
x_{max}	right boundary
y_{min}	bottom boundary
y_{max}	top boundary



3. Types of Line Clipping

There are two commonly used line clipping algorithms :

(i) Cohen-Sutherland Line Clipping

(ii) Liang-Barsky Line Clipping

4. Cohen-Sutherland Line Clipping Algorithm

(i) Region Code

Every point is assigned a 4-bit region code depending on the position of the point w.r.t. the clipping window.

Bit 3	Bit 2	Bit 1	Bit 1
TOP	BOTTOM	RIGHT	LEFT

1 → Point is outside the boundary

0 → Point is inside the boundary

Region Code Chart

	1001 (9)	1000 (8)	1010 (10)
$x < x_{min}$ (Left)	0001 (1)	0000 (0)	0010 (2)
$x > x_{max}$ (Right)	0101 (5)	0100 (4)	0110 (6)

$y < y_{min}$ (Bottom)

(ii) Steps of Algorithm

1. Compute 4-bit region code for both end points P_1 and P_2.
2. If $(code_1 \mid code_2) = 0$ → Both points inside. Accept the line.
3. Else if $(code_1 \,\&\, code_2) \neq 0$ → Both points outside in common region. Reject the line.
4. Else → Line is partially visible. Find the intersection point with the appropriate boundary and replace the outside point by intersection point. Repeat steps 1 to 3.

(iii) Finding Intersection Point

Consider line $P_1(x_1, y_1) - P_2(x_2, y_2)$

Using equation of line in parametric form :

$$x = x_1 + t(x_2 - x_1)$$
$$y = y_1 + t(y_2 - y_1) \qquad 0 \le t \le 1$$

Intersection with boundaries :

- For Left boundary $(x = x_{min})$

  $t = (x_{min} - x_1) / (x_2 - x_1)$

  $y = y_1 + t(y_2 - y_1)$
- For Right boundary $(x = x_{max})$

  $t = (x_{max} - x_1) / (x_2 - x_1)$

  $y = y_1 + t(y_2 - y_1)$
- For Bottom boundary $(y = y_{min})$

  $t = (y_{min} - y_1) / (y_2 - y_1)$

  $x = x_1 + t(x_2 - x_1)$

(iv) Example

Clipping Window : $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$

Line : $P_1(1, 3) - P_2(9, 7)$

Step 1 : Region Codes

$P_1(1, 3) \rightarrow 0001$ (Left)

$P_2(9, 7) \rightarrow 1010$ (Top-Right)

$(code_1 \mid code_2) \neq 0$ and $(code_1 \,\&\, code_2) = 0$ → Partially visible

Step 2 : P_1 is outside (Left). Intersect with Left boundary $x = 2$

$t = (2-1)/(9-1) = 1/8$

$y = 3 + \frac{1}{8}(7-3) = 3 + \frac{1}{2} = 3.5$

New point P_1' (2, 3.5)

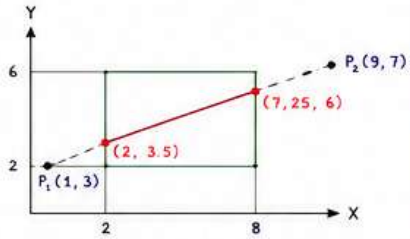
Step 3 : Now P_1' is inside (0000). P_2 is outside (Top-Right). Intersect with Top boundary $y = 6$

$t = (6 - \frac{3}{2}) / (7-3) = \frac{3}{4}$

$x = 2 + \frac{3}{4}(9-2) = 2 + \frac{21}{4} = 7.25$

New point P_2' (7.25, 6)

Clipped Line : P_1' (2, 3.5) – P_2' (7.25, 6)



5. Liang-Barsky Line Clipping Algorithm

Uses parametric form of line :

$$x = x_1 + t(x_2 - x_1)$$
$$y = y_1 + t(y_2 - y_1) \qquad 0 \le t \le 1$$

(i) For each boundary we write inequality in the form :

$t p_k \le q_k$

Boundary	p_k	q_k
Left $(x \ge x_{min})$	$-(x_2 - x_1)$	$x_1 - x_{min}$
Right $(x \le x_{max})$	$(x_2 - x_1)$	$x_{max} - x_1$
Bottom $(y \ge y_{min})$	$-(y_2 - y_1)$	$y_1 - y_{min}$
Top $(y \le y_{max})$	$(y_2 - y_1)$	$y_{max} - y_1$

(ii) Algorithm

1. Initialize $t_E = 0$, $t_L = 1$
2. For each boundary
   - If $p_k = 0$
     - If $q_k < 0$ → Line is outside. Reject.
     - If $q_k \ge 0$ → No effect. Go to next.
   - If $p_k < 0$
     - $t = q_k / p_k$
     - If $t > t_L$ → Reject line.
     - If $t > t_E$ → $t_E = t$
   - If $p_k > 0$
     - $t = q_k / p_k$
     - If $t < t_E$ → Reject line.
     - If $t > t_L$ → $t_L = t$
3. If $t_E \le t_L$ → Line is visible. Clipped line from t_E to t_L.

(iii) Example

Clipping Window : $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$

Line : $P_1(1, 3) - P_2(9, 7)$

p and q values

Left : $p_1 = -8$, $q_1 = -1$

Right : $p_2 = 8$, $q_2 = 7$

Bottom : $p_3 = -4$, $q_3 = 1$

Top : $p_4 = 4$, $q_4 = 3$

$t_E = 0$, $t_L = 1$

For Left : $t = \frac{1}{8} \rightarrow t_E = \frac{1}{8}$

For Right : $t = \frac{7}{8} \rightarrow t_L = \frac{7}{8}$

For Bottom : $t = -\frac{1}{4}$ → No effect

For Top : $t = \frac{3}{4} \rightarrow t_L = \frac{3}{4}$

Since $t_E = \frac{1}{8} \le t_L = \frac{3}{4}$, line is visible.

Clipped End Points :

For $t = \frac{1}{8}$ → (2, 3.5)

For $t = \frac{3}{4}$ → (7, 6)

Clipped Line : (2, 3.5) – (7, 6)

11. Advantages

- ✓ Removes invisible parts and reduces drawing time.
- ✓ Improves display efficiency.
- ✓ Essential for complex scenes.
- ✓ Algorithms are simple and easily implementable.

12. Limitations

- ✗ Depends on window boundaries.
- ✗ Extra computations for intersection points.
- ✗ Not suitable for curved objects (need other techniques).

13. Applications

- Computer graphics and visualization.
- CAD/CAM systems.
- Geographic information systems.
- Robot path planning.
- Image processing and simulations.
- Video games and animations.

14. Key Points

- ✱ Line clipping ensures only visible portion of line is drawn.
- ✱ Cohen-Sutherland uses region codes (4-bit).
- ✱ Liang-Barsky uses parametric form and inequalities.
- ✱ Both algorithms give same clipped line.

15. Summary

Line clipping algorithms are used to determine the visible part of a line lying inside the clipping window. The Cohen-Sutherland algorithm uses region codes and is intuitive, while the Liang-Barsky algorithm is faster and more efficient as it avoids computing intersection points repeatedly. These algorithms are fundamental in the graphics pipeline for realistic display.

Typical Exam Questions

1. Define line clipping. Why is it necessary in computer graphics?
2. Explain Cohen-Sutherland line clipping algorithm with example.
3. Explain Liang-Barsky line clipping algorithm with example.
4. Compare Cohen-Sutherland and Liang-Barsky algorithms.
5. Draw region code diagram for Cohen-Sutherland algorithm.

Key Formulas

- Parametric Form : $x = x_1 + t(x_2 - x_1)$, $y = y_1 + t(y_2 - y_1)$
- Intersection (e.g., Left) : $t = (x_{min} - x_1)/(x_2 - x_1)$, $y = y_1 + t(y_2 - y_1)$
- Inequality Form (Liang-Barsky) : $p_k t \le q_k$

Cohen–Sutherland Line Clipping Algorithm

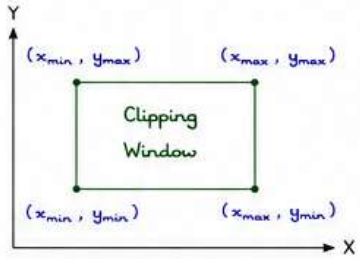
1. Introduction

Line clipping is the process of cutting off the portion of a line which lies outside the clipping window. Cohen–Sutherland line clipping algorithm is a efficient and widely used algorithm based on region codes.

2. Clipping Window

A rectangular clipping window is defined by its boundaries :

- x_{min} – left boundary
- x_{max} – right boundary
- y_{min} – bottom boundary
- y_{max} – top boundary



3. Region Codes (4-bit Code)

The area around the clipping window is divided into 9 regions. Each region is represented by a 4-bit code as follows :

Bit 3	Bit 2	Bit 1	Bit 0
TOP	BOTTOM	RIGHT	LEFT
1001 (9)	1000 (8)	1010 (10)	
0001 (1)	0000 (0)	0010 (2)	
0101 (5)	0100 (4)	0110 (6)	

y ↓ x →

Inside (0000) – Inside the window
Outside – Any non-zero code
Example : Top-Right region = 1010

4. Assigning Region Codes

For a point P(x, y), the 4-bit region code is calculated as :

- Bit 3 (Top) = 1 if $y > y_{max}$ else 0
- Bit 2 (Bottom) = 1 if $y < y_{min}$ else 0
- Bit 1 (Right) = 1 if $x > x_{max}$ else 0
- Bit 0 (Left) = 1 if $x < x_{min}$ else 0

Example :

Clipping Window : $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$

Point (x, y)	Position	Code (Top Bottom Right Left)	4-bit Code
P_1 (1, 3)	Left	0 0 0 1	0001
P_2 (9, 7)	Top-Right	1 0 1 0	1010
P_3 (5, 4)	Inside	0 0 0 0	0000
P_4 (4, 4)	Bottom	0 1 0 0	0100

5. Algorithm

Step 1 : Compute 4-bit region codes for both end points P_1 and P_2.

Step 2 : If ($Code_1$ | $Code_2$) = 0
→ Both points inside. Accept the line.

Step 3 : Else if ($Code_1$ & $Code_2$) ≠ 0
→ Both points outside in same region. Reject the line.

Step 4 : Else
→ Line is partially visible. Find intersection point, replace the outside end point by intersection point and repeat from Step 1.

Logical Operators Used :

| – Bitwise OR
& – Bitwise AND

6. Finding Intersection Point

When a line is partially visible, we need to find intersection with one of the window boundaries.

Line : $P_1(x_1, y_1) - P_2(x_2, y_2)$

We use parametric form :

$$x = x_1 + t(x_2 - x_1)$$
$$y = y_1 + t(y_2 - y_1) \quad ; \quad 0 \le t \le 1$$

Intersection with boundaries :

- Left ($x = x_{min}$) : $t = \frac{x_{min} - x_1}{x_2 - x_1}$, $y = y_1 + t(y_2 - y_1)$
- Right ($x = x_{max}$) : $t = \frac{x_{max} - x_1}{x_2 - x_1}$, $y = y_1 + t(y_2 - y_1)$
- Bottom ($y = y_{min}$) : $t = \frac{y_{min} - y_1}{y_2 - y_1}$, $x = x_1 + t(x_2 - x_1)$
- Top ($y = y_{max}$) : $t = \frac{y_{max} - y_1}{y_2 - y_1}$, $x = x_1 + t(x_2 - x_1)$

7. Example

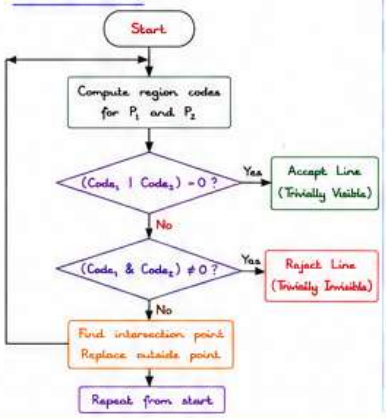
Clipping Window : $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$

Line : $P_1(1, 3) - P_2(9, 7)$

Iteration	P_1 (Code)	P_2 (Code)	Action	Result
1	(1, 3) 0001	(9, 7) 1010	Partially visible	Find intersection with Left boundary
For Left ($x = 2$) : $t = \frac{2-1}{9-1} = \frac{1}{8}$, $y = 3 + \frac{1}{8}(7-3) = 3.5$ New $P_1 = (2, 3.5)$				
2	(2, 3.5) 0000	(9, 7) 1010	Partially visible	Find intersection with Right boundary
For Right ($x = 8$) : $t = \frac{8-2}{9-2} = \frac{6}{7}$, $y = 3.5 + \frac{6}{7}(7-3.5) = 6$ New $P_2 = (8, 6)$				
3	(2, 3.5) 0000	(8, 6) 0000	($Code_1$ | $Code_2$) = 0	Accept the line

Clipped Line : (2, 3.5) – (8, 6)

8. Flowchart



9. Properties

- Based on region codes (4-bit).
- Efficient and fast.
- Suitable for lines.
- Works in both 2D and 3D.

10. Advantages

- ✓ Simple and easy to implement.
- ✓ Requires less computation.
- ✓ Ideal for real-time applications.
- ✓ Avoids unnecessary calculations for completely invisible lines.

11. Limitations

- ✗ Not suitable for curved objects (need polygon clipping).
- ✗ Extra steps required to find intersection points.
- ✗ Works only for rectangular clipping windows.

12. Applications

- CAD/CAM systems.
- Computer graphics and visualization.
- Image processing and display devices.
- GIS and mapping systems.
- Animation and gaming.
- Robot path planning and simulation.

13. Key Points

- Uses 4-bit region codes.
- ($Code_1$ | $Code_2$) = 0 → Accept.
- ($Code_1$ & $Code_2$) ≠ 0 → Reject.
- Else → Partially visible. Clip and repeat.
- Intersection is found using parametric equations.

14. Summary

Cohen–Sutherland line clipping algorithm uses region codes to determine the visible portion of a line inside a rectangular clipping window. It classifies the line as trivially visible, trivially invisible or partially visible and calculates intersection points with window boundaries when required. It is efficient and widely used in computer graphics pipeline.

Typical Exam Questions

1. Define line clipping and explain Cohen–Sutherland line clipping algorithm.
2. Explain region code with example.
3. What are the advantages of Cohen–Sutherland algorithm ?
4. Explain with example how intersection points are found.
5. Compare Cohen–Sutherland and Liang–Barsky algorithms.

Key Formulas

- Parametric Form : $x = x_1 + t(x_2 - x_1)$, $y = y_1 + t(y_2 - y_1)$, $0 \le t \le 1$
- Left ($x = x_{min}$) : $t = \frac{x_{min} - x_1}{x_2 - x_1}$, $y = y_1 + t(y_2 - y_1)$
- Right ($x = x_{max}$) : $t = \frac{x_{max} - x_1}{x_2 - x_1}$, $y = y_1 + t(y_2 - y_1)$
- Bottom ($y = y_{min}$) : $t = \frac{y_{min} - y_1}{y_2 - y_1}$, $x = x_1 + t(x_2 - x_1)$
- Top ($y = y_{max}$) : $t = \frac{y_{max} - y_1}{y_2 - y_1}$, $x = x_1 + t(x_2 - x_1)$

Liang-Barsky Line Clipping Algorithm

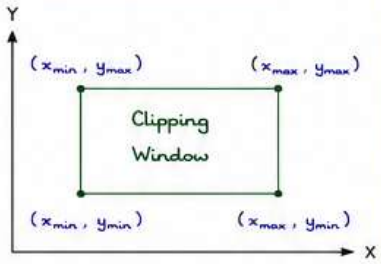
1. Introduction

Liang-Barsky line clipping algorithm is a parametric line clipping technique used to clip a line against a rectangular clipping window. It is faster and more efficient than Cohen-Sutherland algorithm because it requires fewer computations.

2. Clipping Window

A rectangular clipping window is defined by its boundaries.

- x_{min} – left boundary
- x_{max} – right boundary
- y_{min} – bottom boundary
- y_{max} – top boundary



3. Parametric Form of Line

Consider line $P_1(x_1, y_1) - P_2(x_2, y_2)$

Parametric form :

$$x = x_1 + t(x_2 - x_1)$$
$$y = y_1 + t(y_2 - y_1) \quad ; \quad 0 \le t \le 1$$

$t = 0 \rightarrow$ starting point P_1

$t = 1 \rightarrow$ ending point P_2

4. Derivation

Substitute parametric equations in the window inequalities :

$$x_{min} \le x \le x_{max}$$
$$x_{min} \le x_1 + t(x_2 - x_1) \le x_{max}$$
$$y_{min} \le y \le y_{max}$$
$$y_{min} \le y_1 + t(y_2 - y_1) \le y_{max}$$

These can be written in the form :

$$p_i\, t \le q_i \quad , \quad i = 1, 2, 3, 4$$

Where

Boundary	p_i	q_i
Left	$-(x_2 - x_1)$	$x_1 - x_{min}$
Right	$(x_2 - x_1)$	$x_{max} - x_1$
Bottom	$-(y_2 - y_1)$	$y_1 - y_{min}$
Top	$(y_2 - y_1)$	$y_{max} - y_1$

Also, initially

- $t_L = 0$ (entering parameter)
- $t_E = 1$ (leaving parameter)

5. Algorithm

Step 1 : Initialize $t_L = 0$, $t_E = 1$.

Step 2 : For each $i = 1$ to 4, compute p_i and q_i.

Step 3 :

- If $p_i = 0$ and $q_i < 0 \rightarrow$ line is parallel and outside. Reject.
- If $p_i = 0$ and $q_i \ge 0 \rightarrow$ line is parallel and inside. No effect.
- If $p_i < 0 \rightarrow$ potential entering boundary.
  $t = \frac{q_i}{p_i}$; if $t > t_L$ then $t_L = t$
- If $p_i > 0 \rightarrow$ potential leaving boundary.
  $t = \frac{q_i}{p_i}$; if $t < t_E$ then $t_E = t$
- If $t_L > t_E \rightarrow$ line is completely outside. Reject.

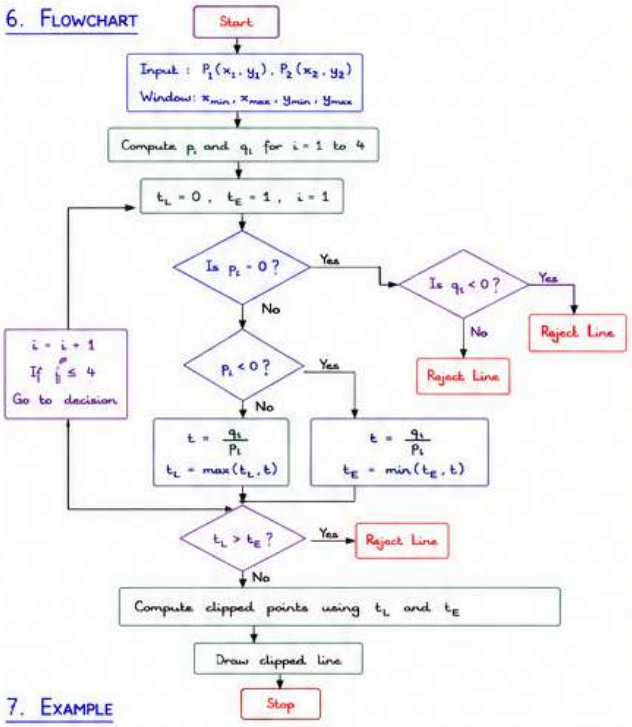
Step 4 : If $0 \le t_L \le t_E \le 1 \rightarrow$ line is visible.
Clipped segment is from t_L to t_E.

Logical Meaning :

$t_L \rightarrow$ the largest entering parameter (first enters window)

$t_E \rightarrow$ the smallest leaving parameter (last leaves window)

6. Flowchart



7. Example

Clipping Window : $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$

Line : $P_1(1, 1) - P_2(9, 7)$

Given: $x_1 = 1$, $y_1 = 1$, $x_2 = 9$, $y_2 = 7$

$dx = x_2 - x_1 = 8$, $dy = y_2 - y_1 = 6$

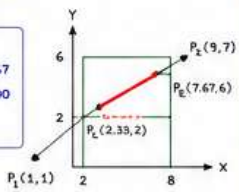
Boundary	p_i	q_i	q_i / p_i	Action	t_L	t_E
Left ($x \ge 2$)	$-dx = -8$	$x_1 - x_{min} = -1$	$(\frac{-1}{-8}) = 0.125$	$p_i < 0 \rightarrow t_L$	0.125	1
Right ($x \le 8$)	$dx = 8$	$x_{max} - x_1 = 7$	$\frac{7}{8} = 0.875$	$p_i > 0 \rightarrow t_E$	0.125	0.875
Bottom ($y \ge 2$)	$-dy = -6$	$y_1 - y_{min} = -1$	$(\frac{-1}{-6}) = 0.1667$	$p_i < 0 \rightarrow t_L$	0.1667	0.875
Top ($y \le 6$)	$dy = 6$	$y_{max} - y_1 = 5$	$\frac{5}{6} = 0.8333$	$p_i > 0 \rightarrow t_E$	0.1667	0.8333

Now, $t_L = 0.1667$, $t_E = 0.8333$ and $0 \le t_L \le t_E \le 1 \rightarrow$ Line is visible.

Clipped Points :

At $t_L = 0.1667$
$x = 1 + 0.1667(8) = 2.333$
$y = 1 + 0.1667(6) = 2.000$
$P_L(2.333, 2.000)$

At $t_E = 0.8333$
$x = 1 + 0.8333(8) = 7.667$
$y = 1 + 0.8333(6) = 6.000$
$P_E(7.667, 6.000)$



8. Steps Summary

- Compute dx and dy.
- Compute p_i and q_i for four boundaries.
- Initialize $t_L = 0$, $t_E = 1$.
- Update t_L and t_E.
- If $t_L > t_E \rightarrow$ reject the line.
- Else $\rightarrow$ compute clipped points using t_L and t_E.

9. Advantages

- ✓ Very efficient (fewer computations).
- ✓ Fast for hardware implementation.
- ✓ Suitable for real-time systems.
- ✓ Handles all line positions easily.

10. Limitations

- ✗ Slightly complex to understand initially.
- ✗ Not directly applicable for curved objects.
- ✗ Needs floating point operations.

11. Applications

- Computer graphics and visualization.
- CAD/CAM systems.
- Image processing.
- Robot path planning.
- GIS and mapping systems.
- Simulation and flight systems.

12. Key Points

- Based on parametric line representation.
- Uses $p_i\, t \le q_i$ form.
- $t_L \rightarrow$ entering parameter (max).
- $t_E \rightarrow$ leaving parameter (min).
- Line visible if $0 \le t_L \le t_E \le 1$.
- More efficient than Cohen-Sutherland algorithm.

13. Summary

Liang-Barsky algorithm is a parametric line clipping algorithm which uses the idea of entering and leaving parameters to clip a line against a rectangular window. It is fast, uses fewer calculations and is widely used in modern graphics pipelines.

Typical Exam Questions

1. Explain Liang-Barsky line clipping algorithm.
2. Derive parametric form of line and inequalities for window.
3. What are t_L and t_E? How are they used?
4. Clip the line $P_1(1, 1) - P_2(9, 7)$ against window $x_{min} = 2$, $x_{max} = 8$, $y_{min} = 2$, $y_{max} = 6$ using Liang-Barsky algorithm.
5. Compare Cohen-Sutherland and Liang-Barsky algorithms.

Key Formulas

- Parametric Line : $x = x_1 + t(x_2 - x_1)$, $y = y_1 + t(y_2 - y_1)$, $0 \le t \le 1$
- Inequalities : $p_i\, t \le q_i$
- Left : $p_1 = -(x_2 - x_1)$, $q_1 = x_1 - x_{min}$
- Right : $p_2 = (x_2 - x_1)$, $q_2 = x_{max} - x_1$
- Bottom : $p_3 = -(y_2 - y_1)$, $q_3 = y_1 - y_{min}$
- Top : $p_4 = (y_2 - y_1)$, $q_4 = y_{max} - y_1$
- Accept line if : $0 \le t_L \le t_E \le 1$

Polygon Clipping Algorithms

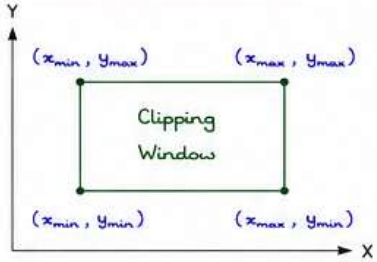
1. Introduction

Polygon clipping is the process of removing the parts of a polygon that lie outside a specified clipping window and retaining only the part visible inside the window.
It is widely used in graphics systems to display only the visible portion of objects.

2. Clipping Window

A rectangular clipping window is defined by its boundaries.

- x_{min} → left boundary
- x_{max} → right boundary
- y_{min} → bottom boundary
- y_{max} → top boundary



3. Polygon Clipping Algorithms

The two most commonly used polygon clipping algorithms are:

- Sutherland – Hodgman Polygon Clipping Algorithm
- Weiler – Atherton Polygon Clipping Algorithm

4. Sutherland – Hodgman Polygon Clipping Algorithm

(ii) Basic Idea

The clipping window is considered as a sequence of four boundaries (Left, Right, Bottom, Top). The input polygon is clipped successively against each boundary.

(ii) Steps

1. Start with the input polygon as the subject polygon.
2. Clip it against the Left boundary.
3. The output becomes the input for the Right boundary.
4. Repeat for Bottom boundary.
5. Finally, clip against the Top boundary.
6. The output after the last step is the clipped polygon.

(iii) Inside Test for Boundaries

Boundary	Inside Condition
Left	$x \geq x_{min}$
Right	$x \leq x_{max}$
Bottom	$y \geq y_{min}$
Top	$y \leq y_{max}$

5. Sutherland – Hodgman Algorithm (Contd.)

(iv) Cases for each edge P_1 (current), P_2 (next)

Case	P_1	P_2	Action
1	Inside	Inside	Output P_2
2	Inside	Outside	Output intersection point with boundary
3	Outside	Inside	Output intersection point and then P_2
4	Outside	Outside	Output nothing

(v) Intersection with boundary

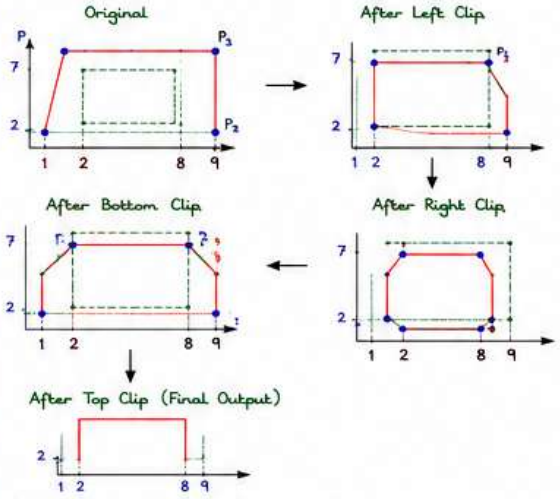
For edge $P_1(x_1, y_1) - P_2(x_2, y_2)$, intersection with a boundary is found using:

Left	$x = x_{min}$; $y = y_1 + (y_2 - y_1)\frac{(x_{min} - x_1)}{(x_2 - x_1)}$
Right	$x = x_{max}$; $y = y_1 + (y_2 - y_1)\frac{(x_{max} - x_1)}{(x_2 - x_1)}$
Bottom	$y = y_{min}$; $x = x_1 + (x_2 - x_1)\frac{(y_{min} - y_1)}{(y_2 - y_1)}$
Top	$y = y_{max}$; $x = x_1 + (x_2 - x_1)\frac{(y_{max} - y_1)}{(y_2 - y_1)}$

(vi) Example

Clipping Window : $x_{min} = 2$, $x_{max} = 8$,
$y_{min} = 2$, $y_{max} = 6$

Input Polygon (in order):
$P_1(1,1)$, $P_2(9,1)$, $P_3(9,7)$, $P_4(1,7)$



6. Weiler – Atherton Polygon Clipping Algorithm

(i) Basic Idea

It works for both convex and concave polygons.
It uses the concept of entering and leaving intersection points.

(ii) Steps

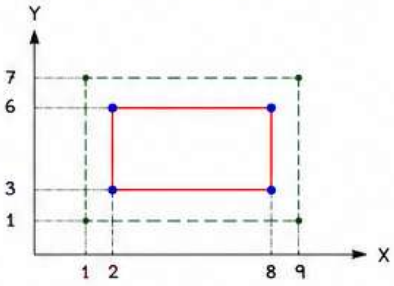
1. Find all intersection points between the subject polygon and the clipping window.
2. Mark each intersection as entering or leaving.
3. Starting from an entering point, follow the subject polygon till the next intersection.
4. Then follow the clipping window boundary till the next intersection.
5. Continue alternating until the starting point is reached.
6. Repeat for all remaining unprocessed entering points.

7. Example (Sutherland – Hodgman)

Clipping Window : $x_{min} = 2$, $x_{max} = 8$,
$y_{min} = 2$, $y_{max} = 6$

Input Polygon :
(1,3), (9,3), (9,7), (1,7)

Final Clipped Polygon (after 4 boundaries):
(2,3), (8,3), (8,6), (2,6)



8. Applications

- Computer graphics and visualization.
- CAD / CAM systems.
- Image processing and pattern recognition.
- Geographic information systems.
- Simulation and gaming.
- Printing and plotting.

9. Advantages

- ✓ Increases display efficiency.
- ✓ Removes invisible parts.
- ✓ Simple and easy to implement (Sutherland – Hodgman for rectangular windows).
- ✓ Weiler – Atherton works for any general polygon.

10. Limitations

- ✗ Sutherland – Hodgman works only for convex clipping window.
- ✗ Weiler – Atherton is complex and requires more memory.
- ✗ More intersection calculations for complex polygons.

11. Summary

Polygon clipping is used to display only the visible portion of polygons inside the clipping window. The Sutherland – Hodgman algorithm is simple and efficient for convex clipping windows, while the Weiler – Atherton algorithm handles concave polygons and concave clipping windows.

Typical Exam Questions

1. Explain polygon clipping and its applications.
2. Describe Sutherland – Hodgman polygon clipping algorithm with neat example.
3. Explain Weiler – Atherton polygon clipping algorithm.
4. Compare Sutherland – Hodgman and Weiler – Atherton algorithms.
5. Find the clipped polygon for a given example.

Key Formulas

- Intersection with boundary (using parametric form)
  $x = x_1 + t(x_2 - x_1)$, $y = y_1 + t(y_2 - y_1)$, $0 \leq t \leq 1$
- Left : $x = x_{min}$
- Right : $x = x_{max}$
- Bottom : $y = y_{min}$
- Top : $y = y_{max}$