

As per the New Revised Syllabus (REV- 2019 'C' Scheme)
of Mumbai University w.e.f. academic year 2020-21

Computer Graphics

(Code : CSC305)

Semester III

Computer Engineering / Computer Science and Engineering /

Artificial Intelligence & Data Science / Machine Learning / Cyber Security /

Internet of Things (IoT) / Data Engineering / Data Science /

Internet of Things and Cyber Security Including Block Chain Technology /

Computer Science Design

Dr. Mahesh M. Goyani



- Solved Latest University Question Papers.
- Model Question Paper for University Examination.

COMPUTER GRAPHICS

(Code : CSC305)

380

Semester III - (Mumbai University)

Computer Engineering / Computer Science and Engineering / Artificial Intelligence & Data Science / Machine Learning / Cyber Security / Internet of Things (IoT) / Data Engineering / Data Science / Internet of Things and Cyber Security Including Block Chain Technology / Computer Science Design

**Strictly as per the New Revised Syllabus (Rev – 2019 ‘C’ Scheme) of
Mumbai University w.e.f. academic year 2020-2021
(As per Choice Based Credit and Grading System)**

Dr. Mahesh M. Goyani

Ph.D., B.A., M.E., B.E.

Assistant Professor,

*Department of Computer Engineering,
Government Engineering College, Modasa
Gujarat, India*

 **Tech Knowledge**TM
Publications

M0148C Price ₹ 425/-



COMPUTER GRAPHICS (CSC305)

Dr. Mahesh M. Goyani

(Semester III - Computer Engineering / Computer Science and Engineering / Artificial Intelligence & Data Science / Machine Learning / Cyber Security / Internet of Things (IoT) / Data Engineering / Data Science / Internet of Things and Cyber Security Including Block Chain Technology / Computer Science Design (Mumbai University))

Copyright © by Author. All rights reserved. No part of this publication may be reproduced, copied, or stored in a retrieval system, distributed or transmitted in any form or by any means, including photocopy, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

This book is sold subject to the condition that it shall not, by the way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior written consent in any form of binding or cover other than which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above.

First Printed in India : August 2015 (Mumbai University)

First Edition : August 2020

Second Revised Edition : July 2021

Third Revised Edition : July 2022

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

ISBN: 978-93-89889-99-4

Published By

TECHKNOWLEDGE PUBLICATIONS

Printed @

37/2, Ashtavinayak Industrial Estate,
Near Pari Company,
Narhe, Pune, Maharashtra State, India.
Pune - 411041

Head Office

B/5, First floor, Maniratna Complex, Taware Colony,
Aranyeshwar Corner, Pune - 411 009.
Maharashtra State, India

Ph : 91-20-24221234, 91-20-24225678.

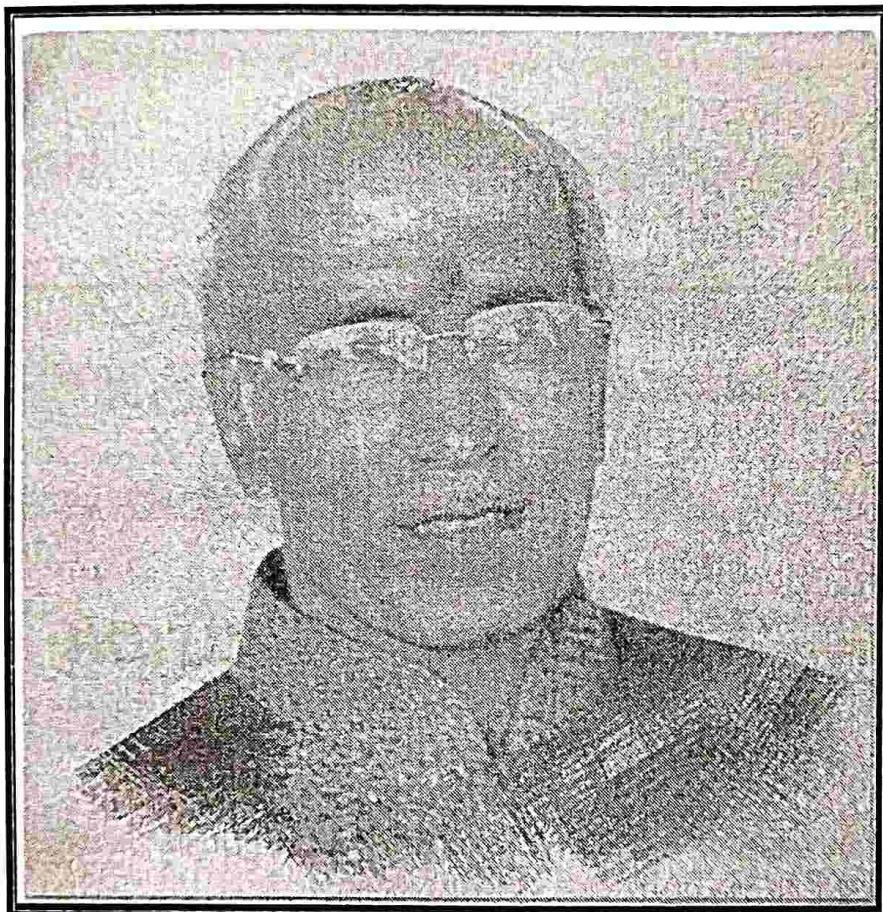
Email : info@techknowledgebooks.com,

Website : www.techknowledgebooks.com

Subject Code : CSC305

Book Code : MO148C

We dedicate this Publication soulfully and wholeheartedly, in loving
memory of our beloved founder director,
Late Shri. Pradeepji Lalchandji Lunawat, who will always
be an inspiration, a positive force and strong support behind us.



“My work is my prayer to God”

- Lt. Shri. Pradeepji L. Lunawat

**Soulful Tribute and Gratitude for all Your
Sacrifices, Hardwork and 40 years of Strong Vision...**

Dedicated to.....

My Beloved Daughters

Kavya and Mausam

The musical lyrics of my life

PREFACE

My dear students,

Computer Graphics is a very dynamic and rapidly growing area amongst others in Computer Science. Entertainment, education, animation, film industry, advertisements, presentations, virtual reality, computer aided design and many other fields rely extensively on computer graphics, owing to its importance Computer Graphics has become an indivisible part not only in academic but in professional ventures as well.

The book "Computer Graphics" has been written according to the new syllabus of Mumbai University for students pursuing B.E., with a vision to provide students the necessary resources for the formation of a strong base in Computer Graphics. This book discusses the concepts of Computer Graphics with self-explanatory diagrams and examples. Care has been taken to provide the reader with an understanding of the principles of Computer Graphics along with their implementation and applications. Numerous examples have been covered to allow the readers an even better understanding of the concepts.

The book has been written in a lucid manner and the technical jargon has been simplified so as to make it comprehensible by people from all disciplines, the questions from previous Mumbai University examinations have been included at the end of every chapter and their solutions are covered within the text. This facilitates the students to prepare a chapter or a concept from an examination point of view as well.

Key features :

- Simplified explanation in lucid manner
- Solved university question papers.
- Algorithms of all problems
- Detail coverage of each and every topic
- Set of self explanatory diagrams
- Wide range of programs



ACKNOWLEDGEMENT

"Computer Graphics" is the result of the numerous discussions. I had in lectures with my students and peers while offering this course at different levels of study. I appreciate all the constructive comments from colleges and doubts raised by students.

I would like to express my gratitude to my colleagues at Government Engineering College, Modasa, for their constant support. They have continuously monitored and motivated my journey throughout this book. My friends and colleagues have been a constant source of inspiration for me. I thank everyone who has contributed to this book directly or indirectly.

I present this book in the loving memory of Late Shri. Pradeepji Lunawat, our source of inspiration and a strong foundation of "TechKnowledge Publications". He will always be remembered in our heart and motivate us to achieve our milestone.

I am thankful to Shri. J. S. Katre, Shri. Shital Bhandari, Shri. Arunoday Kumar and Shri. Chandroday Kumar for the encouragement and support that they have extended. I am also thankful to Seema Lunawat for technology enhanced reading, E-books support and the staff members of TechKnowledge Publications for their efforts to make this book as good as it is. I have jointly made every possible efforts to eliminate all the errors in this book. However if you find any, please let me know, because that will help me to improve further.

Last but not least, I would like to sincerely thank my family for supporting me throughout my venture of writing this book, without them it wouldn't have been possible.

- Dr. Mahesh M. Goyani

SYLLABUS

Mumbai University

Revised syllabus (Rev-2019 'C' Scheme) from Academic Year 2020-21

Course Code	Course Name	Credits
CSC305	Computer Graphics	3

Prerequisites : Knowledge of C Programming and Basic Mathematics.

Course Objectives :

1. To equip students with the fundamental knowledge and basic technical competence in the field of computer graphics.
2. To emphasize on implementation aspect of Computer Graphics Algorithms.
3. To prepare the student for advance areas and professional avenues in the field of Computer Graphics.

Course Outcomes : At the end of the course , the students should be able to

1. Describe the basic concepts of Computer Graphics.
2. Demonstrate various algorithms for basic graphics primitives.
3. Apply geometric transformations on graphical objects.
4. Use various Clipping algorithms on graphical objects
5. Explore 3-D geometric transformations, curve representation techniques and projections methods.
6. Explain visible surface detection techniques and Animation.

Module	Detailed Content	Hours
1.	Introduction and Overview of Graphics System Definition and Representative uses of computer graphics, Overview of coordinate system, Definition of scan conversion, rasterization and rendering. Raster scan & random scan displays, Architecture of raster graphics system with display processor, Architecture of random scan systems. (Refer Chapter 1)	02 hrs.
2.	Output Primitives Scan conversions of point, line, circle and ellipse: DDA algorithm and Bresenham algorithm for line drawing, midpoint algorithm for circle, midpoint algorithm for ellipse drawing (Mathematical derivation for above algorithms is expected) Aliasing, Antialiasing techniques like Pre and post filtering, super sampling, and pixel phasing). Filled Area Primitive : Scan line Polygon Fill algorithm, inside outside tests, Boundary Fill and Flood fill algorithm. (Refer Chapters 2 and 3)	10 hrs.

Module	Detailed Content	Hours
3.	<p>Two Dimensional Geometric Transformations</p> <p>Basic transformations: Translation, Scaling, Rotation Matrix representation and Homogeneous Coordinates Composite transformation Other transformations : Reflection and Shear</p> <p style="text-align: right;">(Refer Chapter 4)</p>	6 hrs.
4.	<p>Two Dimensional Viewing and Clipping</p> <p>Viewing transformation pipeline and Window to Viewport coordinate transformation</p> <p>Clipping operations : Point clipping, Line clipping algorithms : Cohen-Sutherland, Liang : Barsky, Polygon Clipping Algorithms : Sutherland-Hodgeman, Weiler-Atherton.</p> <p style="text-align: right;">(Refer Chapter 5)</p>	7 hrs.
5.	<p>Three Dimensional Geometric Transformations, Curves and Fractal Generation</p> <p>3D Transformations : Translation, Rotation, Scaling and Reflection</p> <p>Composite transformations : Rotation about an arbitrary axis</p> <p>Projections – Parallel, Perspective. (Matrix Representation)</p> <p>Bezier Curve, B-Spline Curve, Fractal-Geometry : Fractal Dimension, Koch Curve.</p> <p style="text-align: right;">(Refer Chapters 6, 7 and 8)</p>	8 hrs.
6.	<p>Visible Surface Detection :</p> <p>Visible Surface Detection : Classification of Visible Surface Detection algorithm, Back Surface detection method, Depth Buffer method, Area Subdivision method</p> <p>Animation : Introduction to Animation, Traditional Animation Techniques, Principles of Animation, Key framing: Character and Facial Animation, Deformation, Motion capture</p>	6 hrs.



Module 1

Chapter 1 : Introduction and Overview of Graphics System

1-1 to 1-28

Syllabus :

Definition and Representative uses of computer graphics, Overview of coordinate system, Definition of scan conversion, rasterization and rendering.

Raster scan & random scan displays, Architecture of raster graphics system with display processor, Architecture of random scan systems.

1.1	Introduction	1-1
1.2	Graphics Primitives	1-2
1.2.1	Lines	1-2
1.2.2	Line Segments.....	1-3
1.2.3	Vectors	1-3
1.3	Applications of Computer Graphics.....	1-4
1.4	Overview of the Coordinate System.....	1-7
1.4.1	Introduction.....	1-7
1.4.2	Cartesian Coordinate System.....	1-8
1.4.3	Polar Coordinate System.....	1-9
1.4.4	Spherical Coordinate System.....	1-10
1.4.5	Cylindrical Coordinate System	1-10
1.5	Basic Terminology.....	1-11
1.5.1	Scan Conversion	1-11
1.5.2	Rasterization.....	1-11
1.5.3	Rendering	1-12
1.5.4	Pixel.....	1-12
1.5.5	Resolution.....	1-13
1.5.6	Aspect Ratio	1-13
1.5.7	Frame Buffer.....	1-14
1.6	Display Devices.....	1-14
1.6.1	Cathode Ray Tube.....	1-15
1.6.2	Raster Scan Display	1-16
1.6.3	Random Scan Display	1-19
1.6.4	Raster Scan vs. Random Scan	1-19
1.6.5	Color CRT Monitors	1-20
1.6.5.1	Beam Penetration Method	1-20
1.6.5.2	Shadow Mask Method	1-20

 Computer Graphics (MU)	1-21
1.6.6 Direct-View Storage Tubes.....	1-22
1.6.7 Flat Panel Display.....	1-23
1.6.7.1 LCD	1-23
1.6.7.2 LED.....	1-24
1.6.7.3 Plasma Panels.....	1-24
1.6.7.4 Electroluminescent Displays.....	1-25
1.6.8 Comparison of CRT, LCD, Plasma and OLED Displays	1-26
1.7 Architecture of Raster Scan Systems	1-26
1.7.1 Video Controller.....	1-28
1.8 Architecture of Random Scan Systems	2.6

Module 2

2-1 to 2-47

Chapter 2 : Output Primitives**Syllabus :**

Scan conversions of point, line, circle and ellipse : DDA algorithm and Bresenham algorithm for line drawing, midpoint algorithm for circle, midpoint algorithm for ellipse drawing (Mathematical derivation for above algorithms is expected)

Aliasing, Antialiasing techniques like Pre and post filtering, super sampling, and pixel phasing).

2.1 Scan Conversion.....	2-1
2.1.1 What is Scan Conversion?	2-1
2.1.2 Line and Line Segment.....	2-2
2.1.3 Qualities of Good Line Drawing Algorithms	2-3
2.2 Line Drawing Algorithms	2-3
2.2.1 DDA Line Drawing Algorithm	2-3
2.2.1.1 Working Mechanism.....	2-4
2.2.1.2 Algorithm	2-7
2.2.1.3 Pros and Cons	2-8
2.2.1.4 Examples.....	2-9
2.2.2 Bresenham Line Drawing Algorithm	2-17
2.2.2.1 Working Mechanism.....	2-17
2.2.2.2 Algorithm	2-19
2.2.2.3 Pros and Cons	2-20
2.2.2.4 Examples.....	2-21
2.2.3 DDA vs. Bresenham Line Drawing Algorithm	2-21
2.3 Circle Drawing Algorithm.....	2-28
2.3.1 Midpoint Circle Drawing Algorithm	2-29
2.3.1.1 Working Mechanism.....	2-29

2.3.1.2	Algorithm	2-31
2.3.1.3	Pros and Cons	2-32
2.3.1.4	Example	2-32
2.4	Midpoint Ellipse Drawing Algorithm	2-35
2.4.1	Working Mechanism	2-35
2.4.2	Algorithm	2-40
2.4.3	Pros and Cons	2-41
2.4.4	Example	2-41
2.5	Aliasing	2-42
2.6	Anti-aliasing	2-43
2.6.1	Pre Filtering	2-43
2.6.2	Post Filtering	2-44
2.6.3	Super Sampling	2-44
2.6.4	Pixel Phasing	2-46

Chapter 3 : Filled Area Primitives

3-1 to 3-16

Syllabus :**Filled Area Primitive :** Scan line Polygon Fill algorithm, Inside outside tests, Boundary Fill and Flood fill algorithm.

3.1	Polygon	3-1
3.1.1	Introduction to Polygon	3-1
3.1.2	Types of Polygon	3-1
3.1.2.1	Convex Polygon	3-1
3.1.2.2	Concave Polygon	3-2
3.1.2.3	Complex Polygon	3-2
3.2	Inside Outside Tests	3-2
3.2.1	Even-Odd Test	3-3
3.2.2	Winding Number Rule	3-4
3.3	Polygon Filling	3-4
3.3.1	Introduction	3-4
3.3.2	Pixel Connectivity	3-5
3.4	Boundary Fill	3-5
3.4.1	Working Mechanism	3-5
3.4.2	Algorithm	3-6
3.4.3	Pros and Cons	3-6
3.4.4	Example	3-7
3.5	Flood Fill	3-8

	Computer Graphics (MU)	4
3.5.1	Working Mechanism.....	3-8
3.5.2	Algorithm	3-8
3.5.3	Pros and Cons.....	3-8
3.6	Optimization	3-9
3.7	Scan Line Fill	3-9
3.7.1	Working Principle	3-9
3.7.2	Algorithm	3-11
3.7.3	Pros and Cons.....	3-12
3.7.4	Example.....	3-12
3.8	Seed Fill vs Scan Line Algorithm	3-16

Module 3**Chapter 4 : Two Dimensional Geometric Transformations**

4-1 to 4-55

Syllabus :

Basic transformations : Translation, Scaling, Rotation, Matrix representation and Homogeneous Coordinates, Composite transformation

Other transformations : Reflection and Shear

4.1	Introduction	4-1
4.2	Matrix Operations	4-2
4.2.1	Matrix Representation.....	4-2
4.2.2	Matrix Properties.....	4-3
4.2.3	Determinant of Matrix	4-3
4.2.4	Multiplying Two Matrices	4-3
4.3	Translation	4-3
4.4	Rotation.....	4-4
4.4.1	Rotation about Origin.....	4-7
4.4.2	Rotation with Respect to Reference Point.....	4-7
4.5	Scaling	4-8
4.5.1	Scaling with Respect to Origin.....	4-12
4.5.2	Scaling with Respect to Reference Point	4-12
4.5.3	Uniform vs. Non-Uniform Scaling	4-13
4.6	Matrix Representation and Homogeneous Coordinates	4-15
4.6.1	Translation.....	4-19
4.6.2	Scaling.....	4-20
4.6.3	Rotation	4-20

4.7	Composite Transformation.....	4-21
4.7.1	Successive Transformations	4-21
4.7.2	Examples.....	4-23
4.8	Reflection.....	4-27
4.8.1	Reflection about X - Axis ($Y = 0$ Line).....	4-28
4.8.2	Reflection about Y - Axis ($X = 0$ Line).....	4-28
4.8.3	Reflection about $X = Y$ Axis.....	4-29
4.8.4	Reflection about $X = -Y$ Axis.....	4-30
4.8.5	Reflection about Origin.....	4-30
4.8.6	Reflection about any Line $y = mx + c$	4-31
4.8.7	Reflection about a Line Parallel to X-Axis.....	4-31
4.8.8	Reflection about a Line Parallel to Y-Axis.....	4-32
4.9	Shearing.....	4-35
4.9.1	X- Direction Shearing	4-35
4.9.2	Shearing about a Line Parallel to X-axis.....	4-36
4.9.3	Y- Direction Shearing	4-38
4.9.4	Shearing about a Line Parallel to Y-axis.....	4-39
4.10	Solved Examples	4-41

Module 4**Chapter 5 : Two-Dimensional Viewing and Clipping**

5-1 to 5-39

Syllabus :

Viewing transformation pipeline and Window to Viewport coordinate transformation

Clipping operations: Point clipping, Line clipping algorithms : Cohen-Sutherland, Liang : Barsky, Polygon Clipping Algorithms : Sutherland-Hodgeman, Weiler-Atherton.

5.1	2D Viewing	5-1
5.1.1	Viewing Transformation Pipeline	5-1
5.1.2	Window to Viewport Coordinate Transformation.....	5-3
5.2	2D Clipping.....	5-5
5.3	Point Clipping	5-6
5.4	Line Clipping	5-7
5.4.1	Cohen - Sutherland Line Clipping Algorithm	5-8
5.4.1.1	Working Mechanism.....	5-8
5.4.1.2	Algorithm	5-10
5.4.1.3	Pros and Cons.....	5-11

 Computer Graphics (MU)	5-11
5.4.1.4 Examples	5-19
5.4.2 Liang - Barsky Line Clipping Algorithm	5-19
5.4.2.1 Working Mechanism	5-20
5.4.2.2 Algorithm	5-21
5.4.2.3 Pros and Cons	5-21
5.4.2.4 Example	5-31
5.5 Polygon Clipping	5-32
5.5.1 Sutherland-Hodgeman Polygon Clipping Algorithm	5-32
5.5.1.1 Working Mechanism	5-34
5.5.1.2 Algorithm	5-34
5.5.1.3 Pros and Cons	5-35
5.5.1.4 Examples	5-36
5.5.2 Weiler - Atherton Polygon Clipping Algorithm	5-36
5.5.2.1 Working Mechanism	5-36
5.5.2.2 Pros and Cons	5-36
5.5.2.3 Examples	5-37

Module 5**Chapter 6 : 3D Transformation**

6-1 to 6-32

Syllabus :

Introduction, Translation, Rotation, Scaling, Reflection, Composite transformations, Rotation about an arbitrary axis

6.1 3D Transformation	6-1
6.1.1 Introduction	6-1
6.1.2 3D Geometry	6-1
6.2 Translation	6-2
6.3 Scaling	6-3
6.3.1 Scaling with Respect to Origin	6-4
6.3.2 Scaling with Respect to Reference Point	6-4
6.4 Rotation	6-4
6.4.1 Rotation About Principal Axis	6-5
6.4.2 Rotation About Line Parallel to Principle Axis	6-6
6.4.3 Rotation About an Arbitrary Axis	6-7
6.5 Reflection	6-8
6.6 Shear	6-13



6.7	Composite Transformations	6-19
6.8	Solved Examples	6-20

Chapter 7 : Projection	7-1 to 7-27
-------------------------------	--------------------

Syllabus :

Projections-Parallel, Perspective. (Matrix Representation)

7.1	Introduction	7-1
7.2	Parallel Projection	7-2
7.2.1	Oblique Parallel Projection	7-3
7.2.1.1	Cavalier Projection	7-4
7.2.1.2	Cabinet Projection	7-5
7.2.2	Orthographic Parallel Projection.....	7-5
7.2.2.1	Multiview Parallel Projection.....	7-5
7.2.2.2	Axonometric Parallel Projection.....	7-6
7.3	Perspective Projection	7-10
7.3.1	Vanishing Point.....	7-12
7.3.2	Single Point Perspective Projection.....	7-13
7.3.3	Two Point Perspective Projection.....	7-14
7.3.4	Three-Point Perspective Projection.....	7-15
7.4	Parallel vs. Perspective Projection.....	7-15
7.5	Orthographic vs. Isometric Projection	7-16
7.6	Solved Examples	7-16

Chapter 8 : Curves and Fractals	8-1 to 8-24
--	--------------------

Syllabus :

Bezier Curve, B-Spline Curve, Fractal-Geometry, Fractal Dimension, Koch Curve

8.1	Curves.....	8-1
8.1.1	Introduction.....	8-1
8.1.2	Interpolation and Approximation.....	8-1
8.1.3	Continuity.....	8-2
8.1.3.1	Geometric Continuity	8-3
8.1.3.2	Parametric Continuity	8-3
8.1.4	Bezier Curve	8-4
8.1.5	B-Spline Curve	8-14
8.1.6	Bezier Vs. B-Spline Curve	8-18

Table of Contents

	Computer Graphics (MU)	8	8-18
8.2	Fractals	8-18
8.2.1	Fractal-Geometry.....	8-19
8.2.2	Fractal Classification	8-20
8.2.3	Fractal Dimension.....	8-22
8.2.4	Koch Curve.....	8-23
8.2.5	Applications

Module 6**Chapter 9 : Visible Surface Detection**

9-1 to 9-9

Syllabus :

Visible Surface Detection : Classification of Visible Surface Detection algorithm, Back Surface detection method, Depth Buffer method, Area Subdivision method

9.1	Introduction	9-1
9.2	Classification of Visible Surface Detection Algorithm	9-1
9.3	Back Surface Detection Method.....	9-2
9.4	Depth Buffer Method.....	9-4
9.5	Area Subdivision Method	9-6

Chapter 10 : Animation

10-1 to 10-9

Syllabus :

Introduction to Animation, Traditional Animation Techniques, Principles of Animation, Key framing : Character and Facial Animation, Deformation, Motion capture

10.1	Introduction	10-1
10.2	Traditional Animation Techniques.....	10-1
10.3	Principles of Animation	10-2
10.4	Keyframing : Character and Facial Animation.....	10-2
10.5	Deformation.....	10-4
10.5.1	Rigid Body Transformation	10-5
10.5.2	Deformation	10-6
10.5.2.1	Parameterized Deformation.....	10-7
10.5.2.2	Lattice Deformation.....	10-7
10.5.2.3	Composite Deformation	10-7
10.6	Motion Capture.....	10-8
>	Model Question Paper (End sem.)	10-8

M-1



Introduction and Overview of Graphics System

Module 1

Syllabus

Definition and Representative uses of computer graphics, Overview of coordinate system, Definition of scan conversion, rasterization and rendering.

Raster scan & random scan displays, Architecture of raster graphics system with display processor, Architecture of random scan systems.

1.1 Introduction

- Computers have been a core part of efficient and economical image generation and manipulation. Advancement in computer technology has made the graphics processing faster and economical.
- Today, the scope of term computer graphics is not limited to what kind of scenes can be displayed or what kind of images we can generate with the help of a computer. In broad mean, computer graphics has covered almost every field, like education, animation, entertainment, film industry, fine arts, gaming, engineering, training, advertisement, medicine, statistical representation and many more.
- **Computer graphics** is a process of generating, manipulating, storing and displaying images.
- A person is more interested to study monthly or annual statistics in the form of graphs, tables or charts, instead of plain boring textual data.
- Computer graphics is the key component for presentation and user interface. Old slow and time-consuming command line interfaces have been replaced by quick, fascinating Graphical User Interface (GUI). GUI provides quick and easy to use interface to the application users.
- Due to inefficient and expensive hardware before a couple of decades, application areas of computer graphics were limited. But with evolution in a digital system and personal computers, there is a flood of inexpensive graphics-based applications and devices.
- With the help of GUI, multiple users can easily control simultaneous applications like text editors, code editors, entertainment tools etc. Computer graphics was initially started with displaying data on the hard copy display devices like plotters, printers and then Cathode Ray Tube (CRT). Today, the applications of computer graphics have been extended to display devices like Light Emitting Diodes (LED), Liquid Crystal Display (LCD), electroluminescent display, flat CRTs etc.
- To generate a realistic scene, computer graphics performs various operations on objects such as translation, rotation, scaling, reflection, shearing, hidden surface removal, texture mapping, shading.
- Operations of computer graphics can be classified as follows :
 1. **Geometry** : How to draw and represent the scene.



2. **Animation** : How to add dynamic content in the scene.
3. **Rendering** : How to add realistic light in the scene.
4. **Imaging** : Image acquisition and image editing.

1.2 Graphics Primitives

- On the monitor, graphics primitives are generated using scan conversion.
- **Scan conversion** is the process of turning 'on' or 'off' the pixels along the boundary of the primitive shape.
- For each pixel on the screen, scan conversion algorithm performs the computation based on the initial parameters of the primitives and determines which pixel is to be turned 'on'
- Few basic primitives are discussed in following subsections.

1.2.1 Lines

- The line is a collection of points along a straight path. Typically, the line is described by its two endpoints (x_1, y_1) and (x_2, y_2) . Intermediate points on the line are calculated by line generation algorithm.

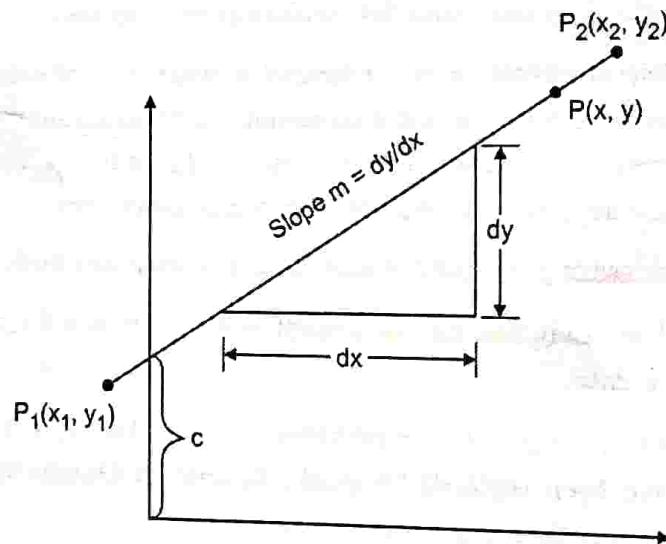


Fig. 1.2.1 : Representation of line

- Given the endpoints $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, any point $P(x, y)$ on the line can be computed using the slope intercept formula as follow.
- It is evident from Fig. 1.2.1 that the slope of the line is,

$$m = \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1}$$

- Let 'c' represents the Y-intercept of the line. Given the x-coordinate, the value of the corresponding y-coordinate of any point is computed as follow.

$$y = mx + c$$



- Lines with different slopes are shown in Fig. 1.2.2.

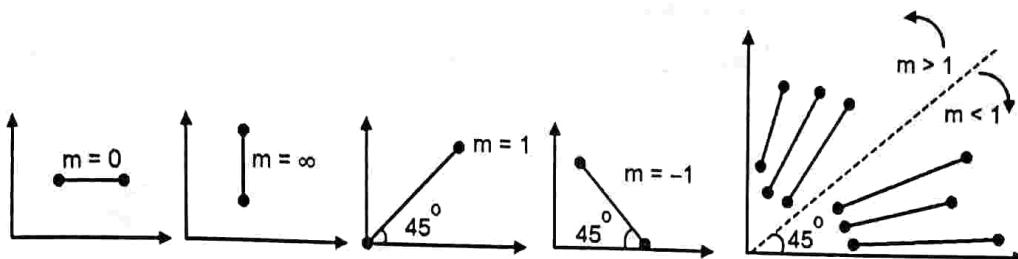


Fig. 1.2.2 : Lines with different slope

1.2.2 Line Segments

- A line segment is the part of the line. Curves are often approximated using small line segments. Almost any shape can be approximated by line segments.
- Like line, a line segment is also bounded by some start and endpoints.

1.2.3 Vectors

- In geometry, a vector is defined by two properties : direction and the magnitude.
- We can represent a vector as a directed line, which grows from tail to head.

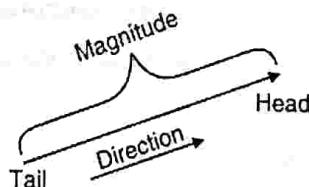


Fig. 1.2.3 : Vector representation

- Two vectors are said to be same if, they have the same direction and magnitude.
- A vector in 2D is defined as $v = ai + bj$, where (i, j) represents the direction and (a, b) represents the magnitude of a vector in x and y direction, respectively.
- Given the two points $A(x_1, y_1)$ and $B(x_2, y_2)$, the vector in the direction from A to B is defined as,

$$v = \vec{AB} = B - A = (x_2 - x_1, y_2 - y_1)$$

- And the gradient of vector \vec{AB} is given as,

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

Ex. 1.2.1 : Compute the direction and magnitude of the vector \vec{AB} given the points $A(2, 4)$ and $B(7, 8)$.

Soln. : The direction of the vector is computed as,

$$v = \vec{AB} = B - A = (x_2 - x_1, y_2 - y_1) = (7 - 2, 8 - 4) = (5, 4)$$

The magnitude of the vector is computed as,

$$\| v \| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = \sqrt{5^2 + 4^2} = \sqrt{41} = 6.4$$

Thus, the vector is pointing in the direction $5i + 4j$ and has magnitude 6.4. The geometric representation of the points and vector is shown in the Fig. P. 1.2.1.

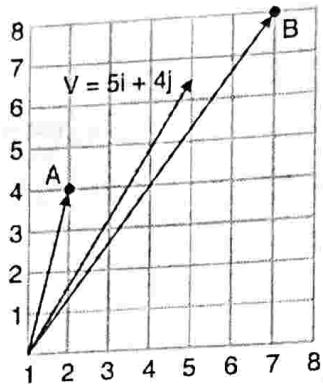


Fig. P. 1.2.1 : Geometric interpretation of vector

1.3 Applications of Computer Graphics

Computer graphics has its root almost everywhere. In this section, we will discuss a few applications which are used in day to day life.

- Computer-Aided Designing :** One of the fundamental applications of computer graphics is designing and modeling of objects, which is known as Computer-Aided Design (CAD). CAD methods are common in designing models of machines, buildings, crafts, computers, automobiles, textiles, and many other products. CAD packages allow designers to make wireframe or solid models of components. Wireframe display gives flexibility to designers to quickly see the effects of changes in shapes and adjust it.

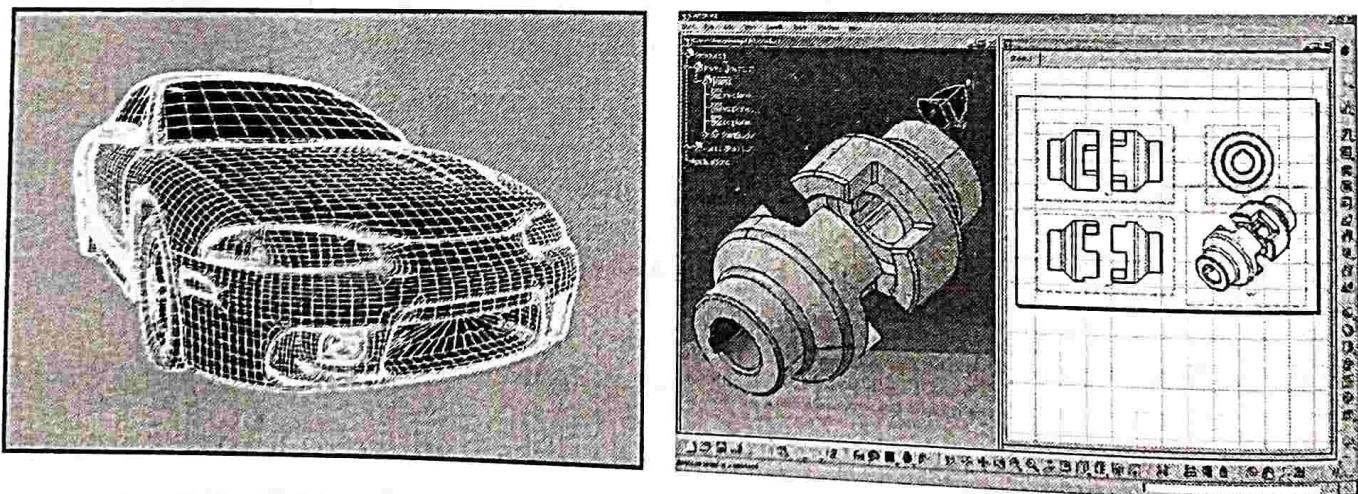


Fig. 1.3.1 : Role of computer graphics in Computer-Aided Designing

Software packages for CAD applications also provide the designer with a multi-window environment which can display different views of objects. Standard shapes for various logical circuits are also often supplied by the electronic circuit design packages. Animations are often used in CAD applications. Real-time wireframe display is useful for testing the performance of a vehicle or system. Also, wireframe displays allow the designer to see into the interior of the vehicle and to watch the behaviour of inner components during motion. Wireframe design of the car and multi-window view of the objects are shown in Fig. 1.3.1.

2. **Virtual reality :** With the help of computer graphics, it is possible to simulate various virtual environments which look seamlessly similar to the real world. Such environments help to train the person without providing them with physical resources. Doctors can be trained to operate a patient without operating on the actual physical body. The pilot can be trained to fly the aircraft without physically seating in the craft. One such vehicle navigation simulator is shown in Fig. 1.3.2. It is also possible to have simulated "walk" in and around the home using virtual reality.



Fig. 1.3.2 : Virtual vehicle navigation system

3. **Presentation graphics :** Another major application of computer graphics is presentation graphics. It is used to produce diagrams, tables and charts for reports or presentations. Presentation graphics are commonly used to summarize research or survey data. 3D graphs are sometimes used to provide a multi-angle analysis of statistical data. They provide a more intuitive and more attractive presentation of data relationships.

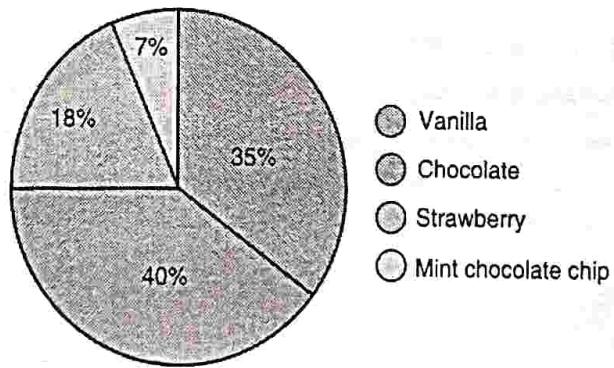


Fig. 1.3.3 : Graphical representation of the survey

4. **Computer art :** Computer graphics play a major role in fine art and commercial art. Various computer methods, symbolic packages, software packages, CAD packages, etc. are used by the designer to generate computer art. Using such packages, the artist can easily design shapes and can add motion.

Morphing is the common graphics processing method used in animation. It is a process of transforming one object into another. Morphing is being used in movies to transform one shape to others, e.g. human to tiger, car or any random shape etc.



Fig. 1.3.4 : Morphing

5. **Entertainment** : Computer graphics are commonly used in film industries for making animation and cartoon movies, for providing special effects to the scene, television shows etc. Sometimes the character itself is displayed as graphics object, and sometimes it is combined with the real scene and synthesized objects.



Fig. 1.3.5 : Use of computer graphics in animation and movie

6. **Education and Training** : Another promising application area of computer graphics is education and training. Nowadays in the market, many educational toolkits are available for child education. They can learn through videos and animations. Thousands of simulation software are available to understand the behaviour of certain processes or environment or response of the system. Training a person for vehicle driving using such simulators saves his time and efforts.

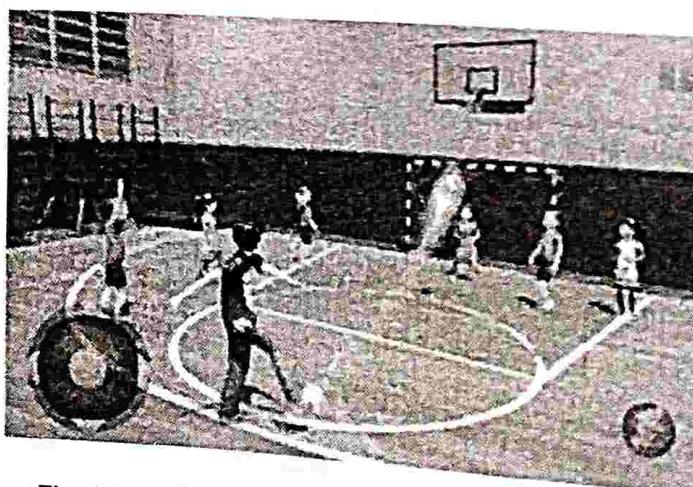


Fig. 1.3.6 : Scope of computer graphics in training

7. **Visualization :** Data produced from many research applications is too big to analyze on paper. Representing data in form of graphs, charts or some other visual form may help to understand the results in quick time. Colour coding can also provide a clue to understanding the data. For example, plain satellite image may not be as useful as if it is segmented using different colors i.e. green indicating forest region, blue indicating water region, black indicating mountains and roads, grey indicating soil etc.

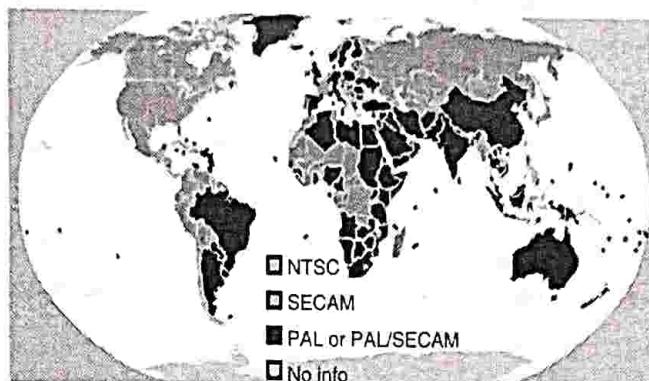


Fig. 1.3.7 : Satellite images with augmented data

8. **Graphical User Interface :** The old computer systems were mostly command-based. The user had to write commands to perform any task. But with the introduction of computer graphics, things have changed. People get more interested in the graphical user interface. Nobody would like to remember or write commands. GUI provides an easy to use interface, with more accuracy and quick access time. GUI packages allow to design and interact with multiple windows like menu-based selection, mouse and keyboard-driven commands, zooming of objects etc. simultaneously.



Fig. 1.3.8 : Graphics User Interface

1.4 Overview of the Coordinate System

1.4.1 Introduction

- The coordinate system determines how and where to display objects. Multiple coordinate systems can work in parallel.
- Objects in real world are represented using the world coordinates whereas an object displayed on a display device is specified in device coordinates. Viewing coordinate system specifies the position of the viewer.

- Generally, graphics packages are designed to be used with a Cartesian coordinate system. Special-purpose packages may allow use of many other coordinate systems.
- In general, several different Cartesian reference frames are used to construct and display a scene. We can create objects, such as a pen or chair, with their actual dimensions, called **modelling coordinates**, or local **coordinates or master coordinates** i.e. (x_{mc}, y_{mc}) . Once the object is designed, we can place it into an appropriate position in the scene using **world coordinates** i.e. (x_{wc}, y_{wc}) .
- The scene may be displayed on different devices, so it must be mapped to the proper dimension to render it. World coordinates are then represented in **normalized coordinate** (x_{nc}, y_{nc}) , usually between 0 and 1. This makes the system independent of the various display devices.
- These normalized coordinate can be easily scaled to fit on any size of the display by multiplying it with a proper scaling factor. This new scaled coordinate is known as **device coordinate** (x_{dc}, y_{dc}) . Fig. 1.4.1 shows sequence of coordinate transformations.

$$(x_{mc}, y_{mc}) \rightarrow (x_{wc}, y_{wc}) \rightarrow (x_{nc}, y_{nc}) \rightarrow (x_{dc}, y_{dc})$$

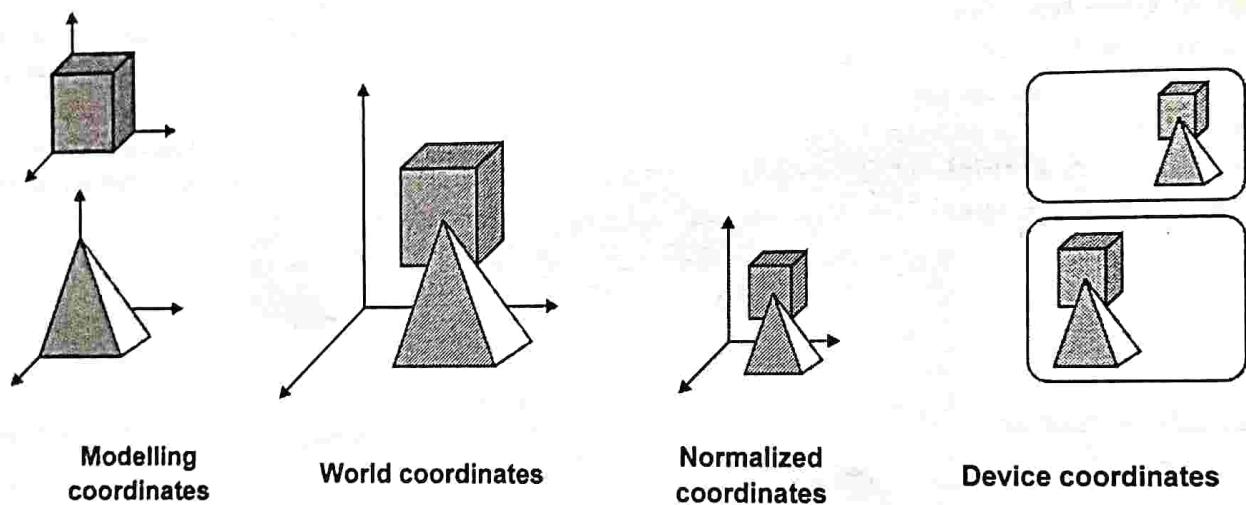


Fig. 1.4.1 : Coordinate transformation

- If normalized coordinates are mapped into a square area of the output device, then only appropriate scale and aspect ratio are maintained. On non-square display device, the shape of object deforms.

1.4.2 Cartesian Coordinate System

- The cartesian coordinate system uses spatial location in a plane or space to represent point uniquely. To represent a point in the plane, we need two-dimensional coordinate systems, which represents point using (x, y) pair.
- To represent a point in the space, we need a three-dimensional coordinate system, which represents point using (x, y, z) triplet. Axes in the Cartesian coordinate system are orthogonal (perpendicular) to each other.
- Various shapes like circle, ellipse, line etc. are represented in a Cartesian coordinate system using a mathematical formula. The cartesian coordinate system is also known as a **rectangular coordinate system**.

- The cartesian coordinate system specifies the physical dimensions of objects. By conventions, X, Y and Z-axis represent the length, height and depth, respectively. Fig. 1.4.2 describes 2D and 3D Cartesian coordinate systems.

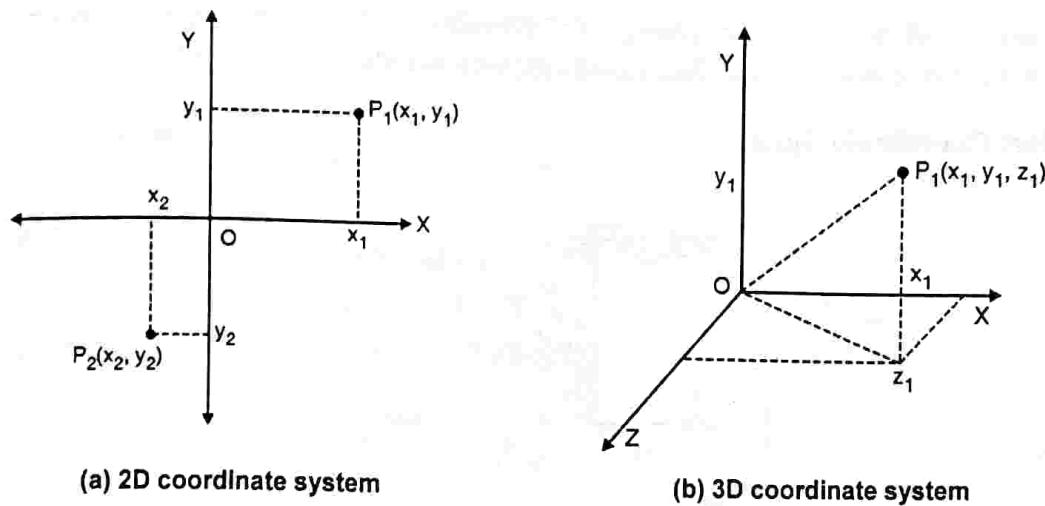
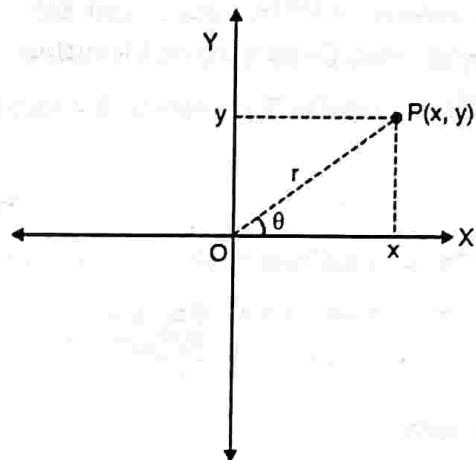


Fig. 1.4.2 : Cartesian coordinate systems

1.4.3 Polar Coordinate System

- The polar coordinate system is also known as a **circular coordinate system**. It uses the distance from origin and angle with the horizontal axis to define the coordinates of the point. Representation of point $P(x, y)$ in polar system is shown in Fig. 1.4.3.

Fig. 1.4.3 : Polar representation of point (x, y)

- Distance of point from the origin is called a **radius**. The anti-clockwise direction is considered as a positive angle. The angle is measured either in degree(0° to 360°) or in radians (0 to 2π). Fig. 1.4.3 shows the polar representation of point $P(x, y)$ in the 2D polar coordinate system.
- From Fig. 1.4.3, point P makes an angle θ with X-axis and the distance of point P from the origin is r. From the trigonometric rule, coordinates of P is given as,

$$\cos \theta = \frac{x}{r}$$

$$\sin \theta = \frac{y}{r}$$

$$\text{So, } x = r \cos \theta$$

$$y = r \sin \theta$$

- This coordinate system is useful in describing motion in objects and the human body. Objects and body are made up of joints and motions are best described by polar coordinates.

1.4.4 Spherical Coordinate System

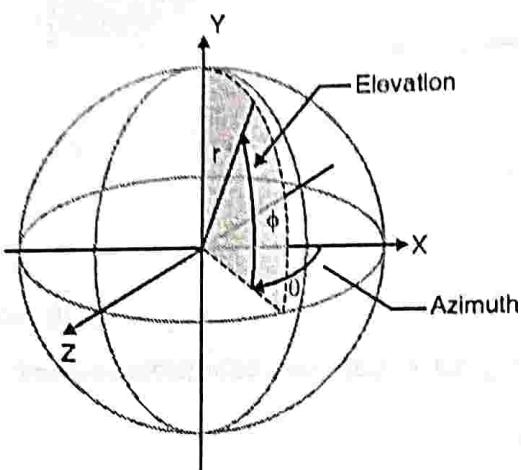


Fig. 1.4.4 : Spherical coordinate system

- The 3D polar coordinate system is also known as a spherical coordinate system.
- The coordinate of the point is parameterized by two angles and distance from the origin. Angle with the The X-axis is called **Azimuth** and angle with Y-axis is called **Elevation**.
- Fig. 1.4.4 shows the spherical coordinate system. The angle in the counter-clockwise direction is considered positive.
- Spherical coordinates are given as,

$$x = r \sin \theta \cos \phi, \quad 0 \leq \theta \leq \pi, \quad 0 \leq \phi \leq 2\pi$$

$$y = r \sin \theta \sin \phi, \quad 0 \leq r \leq \infty$$

$$z = r \cos \theta$$

1.4.5 Cylindrical Coordinate System

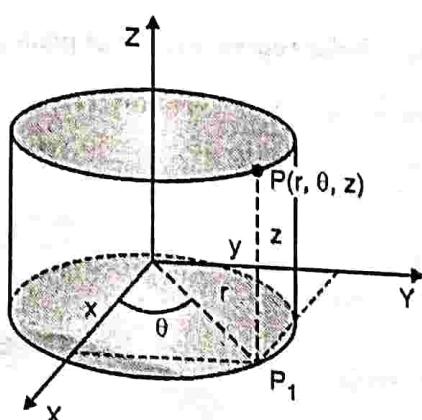


Fig. 1.4.5 : Cylindrical coordinate systems

- Cylindrical-coordinate system is an extension of the circular polar system with the added height of the point from the XY plane. Fig. 1.4.5 shows the representation of the cylindrical coordinate system.
- Azimuth angle with X-axis is 0 and height of point from the surface is z. Point P is represented using these parameters as $P(r, \theta, z)$, where r is the distance between origin and projection of P in the XY plane.

1.5 Basic Terminology

1.5.1 Scan Conversion

- **Scan conversion** is the process of representing continuous graphics as a collection of discrete pixels. Each pixel on the monitor screen is either turned *on* or *off* according to the picture definition stored in the frame buffer.
- Scan conversion is widely used in CRT, Flat-panel, TV set and other types of display devices to display.
- Implementation of such algorithm varies from one computer system to others. Some algorithms are implemented in software, some are implemented in firmware and some are the combination of hardware, software and firmware.
- Algorithms implemented in hardware scan converts the objects quicker than software.
- Application program takes these parameters as input and generates all the pixels representing the shape. This process is called scan conversion.
- All the points on a line can be generated using DDA or Bresenham's algorithm for given endpoints.
- Most of the monitors and television sets support raster scan display. In raster scan devices, picture definition is stored as a discrete pixel value. The entire scene is defined by a set of intensity values for each pixel on the screen.
- Computed pixel values may be a fraction, so it needs to round off to show it on the monitor screen.
- As shown in Fig. 1.5.1, computed pixel values (10.8, 10.8), (12.6, 11.7) and (13.3, 10.4) are rounded to nearest integer values (11, 11), (13, 12) and (13, 10).

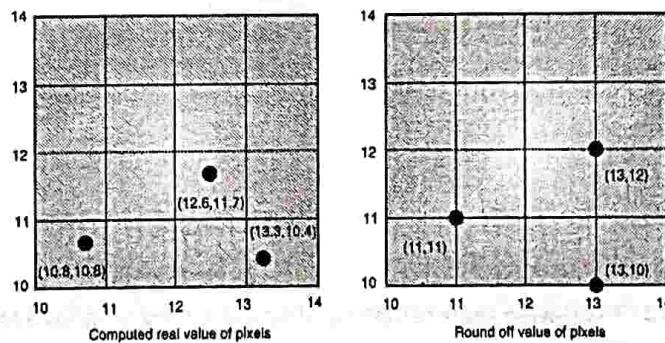


Fig. 1.5.1 : Conversion from actual value to rounded value

1.5.2 Rasterization

- **Rasterization** is the process of converting vector graphics into raster graphics. More formally, it converts the shapes into a set of pixels and displays it on the monitor screen or printer.



- Process of rasterization is depicted in the following Fig. 1.5.2.

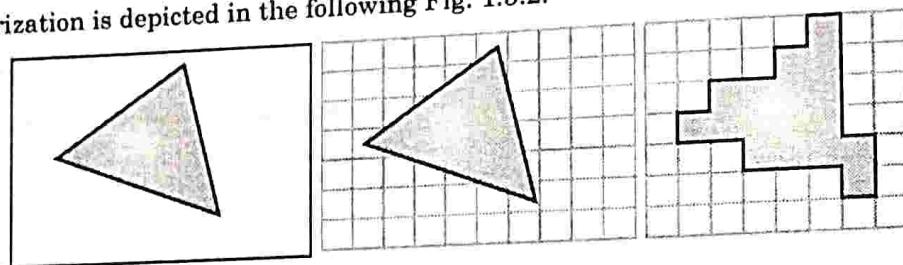


Fig. 1.5.2 : Rasterization of vector scene

1.5.3 Rendering

- Rendering refers to the process of generating photorealistic image from 2D or 3D models using some computer program.
- The scene is stored using specific data types in the scene file. Rendering program retrieves the information like vertex position, viewpoint, color, texture, lighting, shading etc. and render the scene on the monitor screen.
- Thus, the term rendering is analogous to the 'artist's rendering' of the scene.
- Rendering is mostly used in video games, simulators, special effects and visual designs.

1.5.4 Pixel

- The pixel is the smallest addressable unit of an image/screen. It is also known as pel or picture element. Real-world objects are continuous and hence they have infinite samples.
- When they are converted into an image and displayed on the monitor screen, only finite samples are taken into consideration. Ultimately, the object on the computer screen is a collection of pixels.
- Like the digital image, the monitor screen is also divided into a grid, consisting of 'n' rows and 'm' columns. Each cell on this grid is also referred to as a pixel. Each grey/black cell in Fig. 1.5.3 corresponds to one pixel.

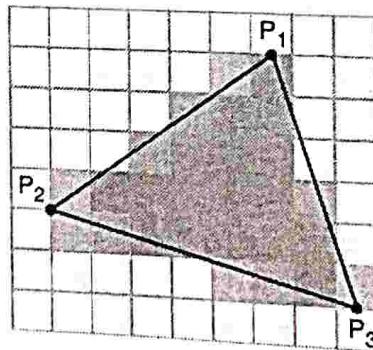


Fig. 1.5.3 : Pixel-wise representation of shape on the monitor screen

- On zooming in the image, we can get the better idea of the pixel. Each square in Fig. 1.5.3 represents one pixel.
- Pixels contained within the object (triangle here) are displayed with full intensity. If the pixel is partially covered by the object boundary, the pixel is illuminated with proportional intensity. The pixel with more area inside the triangle will have a higher intensity than the pixel with less area inside the object boundary.

- This is sufficient to clarify the difference between the object drawn on paper and displayed on the monitor. Objects drawn on paper have smooth boundary whereas objects displayed on the monitor have a zig-zag boundary.
- Pixels outside the object will not be illuminated.
- For a grayscale image, pixel value varies from 0 to 255, where 0 represents the black color and 255 represents the white color. In between range represents the grey shades with varying intensities.
- For a color image, each pixel is a triplet of color(R, G, B) consisting proportion of red, green and blue color. The mixture of these three components determines the final color of the pixel on the monitor screen. Value of each component varies from 0 to 255, resulting in $255 \times 255 \times 255$ i.e. approximately 17 million different shades of color.

1.5.5 Resolution

Q. Explain term resolution with suitable example.

MU - Dec. 19, 2 Marks

- The monitor screen is a grid of pixels. The maximum number of pixels that can be displayed without overlap on the monitor screen is called **screen resolution**.
- Typical resolution on high-quality systems is 1280 by 1024. Such high-resolution systems are often referred to as **high-definition systems**.
- Images on the monitor screen are described by its resolution. Image with resolution 800×600 means that the image has 800 pixels in each row and 600 pixels in each column. Image contains total $800 \times 600 = 4,80,000$ pixels.
- The number of pixels defines the actual number of samples are used from the real-valued scene. Better the resolution, higher the quality of the displayed scene.
- As resolution decreases, fewer samples will be captured and images quality gets poor. Image with resolution 256×256 , 64×64 and 16×16 are shown in Fig. 1.5.4.



Fig. 1.5.4 : Images with different resolution

1.5.6 Aspect Ratio

Q. Define Aspect ratio

MU - Dec. 19, 2 Marks



- Another property of video monitor is the aspect ratio, which is the ratio of the number of vertical pixels to the number of horizontal pixels required to draw the same length of a line in both the direction.
- An aspect ratio of 9/16 means that a vertical line plotted with 9 points has the same length as a horizontal line plotted with 16 points.
- Generally, the distance between two vertical pixels is more than a distance between two horizontal pixels.
- Different aspect ratios are used in different devices. For example, 20th-century monitors were using an aspect ratio of 4 : 3, widescreen televisions use the aspect ratio of 16 : 9, ultra-widescreen devices use the aspect ratio of 64 : 27.

1.5.7 Frame Buffer

- Raster system displays the picture by drawing pixel in a row by row from left to right. Picture definition is stored in a memory called the **refresh buffer** or **frame buffer**.
- Scene to be displayed is first loaded into a frame buffer in the form of intensity values. Ideally, the size of the frame buffer is the same as screen resolution. Intensity value from the top-left location of the frame buffer is retrieved and painted at a top-left location on the screen. The second pixel of the same row is painted soon after that and this process continues.
- Frame buffer in raster display stores individual pixels of the scene, whereas in case of random scan display, frame buffers store commands for drawing the scene.
- Suppose a system has resolution 1280×1280 and it supports 24 bits per pixel. The memory requirement for the frame buffer is;

$$\text{Number of pixel} = 1280 \times 1280 = 16,38,400$$

$$\text{Number of bits} = \text{Number of pixels} \times \text{Number of bits/pixel}$$

$$= 16,38,400 \times 24 = 3,93,21,600 \text{ bits}$$

$$= 39321600/8 = 49,15,200 \text{ Bytes} \quad (\because 1 \text{ Byte} = 8 \text{ bits})$$

$$= 49,15,200/1024 = 4800 \text{ KB} \quad (\because 1 \text{ KB} = 1024 \text{ Bytes})$$

$$= 4800/1024 = 4.69 \text{ MB} \quad (\because 1 \text{ MB} = 1024 \text{ KB})$$

- Thus, the system having screen resolution 1280×1280 and supporting 24 bits per pixel requires frame buffer of size 4.69 MB.
- The frame buffer is known as a **bitmap** if it uses one bit per pixel. And it is known as **pixmap** if it uses multiple bits per pixel.
- A bitmap is used in the bi-level system while pixmap is used in a system supporting multiple colors.
- Sometimes the additional alpha channel is retained to adjust the transparency of a pixel.

1.6 Display Devices

Display devices are output devices used to display information such as the monitor, television set. The information displayed on video display devices is transient. It is displayed for some time and then erased. In the following section, we will discuss few popular display devices.

1.6.1 Cathode Ray Tube

Q. Draw the diagram of CRT and explain its working.

MU - Dec. 19, 5 Marks

- Cathode ray tube (CRT) is a common technology for traditional monitors and television sets. Functional diagram of CRT is depicted in Fig. 1.6.1.

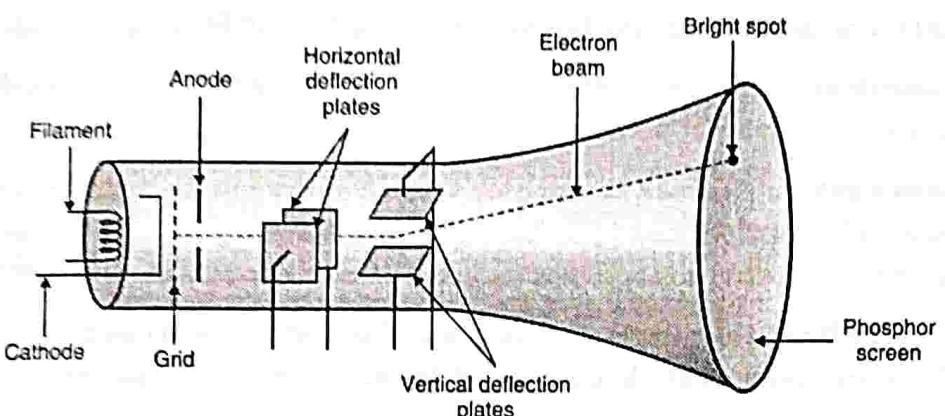


Fig. 1.6.1 : Functional diagram of CRT

- When current is supplied to the filament, electrons are produced and they get dispersed all around. Electrons generated from electron guns fly in arbitrary directions. Focusing anode focuses all dispersed electrons in a beam called an *electron beam*. For higher precision, additional focusing hardware is used to keep the beam in focus at all screen positions.
- The monitor screen is generally curvature in shape, the distance of all screen pixels is not the same from the electron beam. Electron beam focuses properly at the center of the screen. As the beam moves away from the center i.e. near to the edges, the scene on screen become blurred. To overcome this effect, the system can adjust the focusing according to the screen position of the beam.
- All those fired electrons form a beam and start flowing towards phosphorus screen, which is exactly behind the glass of the monitor screen. A positively charged anode, next to the grid, provides acceleration to these electrons. Electrons are negatively charged and an anode is positively charged, so they get attracted towards the anode and get acceleration.
- The beam of electrons in acceleration collides with phosphorus screen and their kinetic energy is absorbed by the screen. This action moves the electrons into their higher quantum energy level and electrons become unstable. As nature, electrons release the energy and try to become stable. This energy appears as an illuminated bright spot on the screen, which we call *pixel*.
- In RGB monitors, there are three layers of phosphorus behind screen, Red, Green and Blue. So color and intensity of illuminated pixel depend on how deep electron beam has penetrated particular layer and what amount of electrons were generated from electron guns.
- When electrons become stable, they stop releasing energy and after a few milliseconds pixel goes off from the screen. To persist the scene on the screen, it is essential to keep redrawing all the pixels repeatedly on the monitor screen. This process is called *refreshing* and such a display system is called *refresh CRT*.



- Apart from color, the main difference between phosphors is their persistence rate. Persistence rate defines how long pixels continue to emit light after the CRT beam is removed. The time it takes the emitted light to decay to one-tenth of its original intensity is called **persistence** of phosphor. If phosphor has a lower persistence rate, it requires higher refresh rates to maintain a picture on the screen without flicker.
- Animation requires quick erase and redrawing of the scene. So low presentence phosphor is preferred in such applications. High-persistence phosphor is useful for displaying highly complex and static pictures. Usually, graphics monitors use phosphorus with persistence in the range from 10 to 60 microseconds.
- Brightness or intensity of a pixel is directly proportional to the number of electrons colliding on phosphorus layers. We can control the number of electrons by controlling the width of the control grid.
- Control grid acts as a gate for electrons. On applying a high negative voltage, control grid shuts off the beam by repealing and not allowing electrons to pass through it means the gate is closed for electrons. On applying a smaller negative voltage, control grid open ups the gate and more electrons can pass through it.
- If the movement of the flowing electron beam is not controlled, it continuously keeps hitting the center pixel of the screen. The energy released by electrons cumulated at a single point, and very soon the phosphorous layer will burn out at that location.
- The position of the colliding beam is controlled either by electric fields or by magnetic fields. Cathode-ray tubes are constructed with magnetic deflection coils which are mounted outside of the CRT.
- Electrostatic deflection system uses two pairs of parallel plates mounted inside the CRT(as shown in Fig. 1.6.1). In both cases, a pair of coils are used to control the movement of the electron beam in the horizontal and vertical direction.
- One set of plate controls the horizontal deflection of the beam and other sets of plate control the vertical deflection. Using such a focusing system, we can make an electron beam to collide with each pixel location on the screen to illuminate the entire scene at a time.
- A maximum number of pixels that can be displayed without overlap on a screen is called **screen resolution**. Typical resolution on high-quality systems is 1280 by 1024. Such high-resolution systems are often referred to as **high-definition systems**.
- Another property of video monitor is the **aspect ratio**, which is the ratio of the number of vertical pixels to the number of horizontal pixels required to draw the same length of a line in both the direction. An aspect ratio of 9/16 means that a vertical line plotted with 9 points has the same length as a horizontal line plotted with 16 points. Generally, the distance between two vertical pixels is more than the distance between two horizontal pixels.

1.6.2 Raster Scan Display

- We can consider the screen picture as a two-dimensional grid of pixels.
- Raster system displays the picture by drawing pixel in a row by row from left to right. Picture definition is stored in a memory called the **refresh buffer or frame buffer**.
- Scene to be displayed is first loaded into a frame buffer (a reserved memory for storing the scene to be displayed on a monitor screen) in the form of intensity values.

- Ideally, the size of the frame buffer is the same as screen resolution. Intensity value from the top-left location of the frame buffer is retrieved and painted at a top-left location on the screen. The second pixel of the same row is painted soon after that and this process continues.
- After plotting all pixels in the first row, the electron beam moves to the first pixel of the second row and paints all pixels in the second row on-screen by retrieving intensity values in horizontal order. Fig. 1.6.2 describes the working principle of the raster scan system.

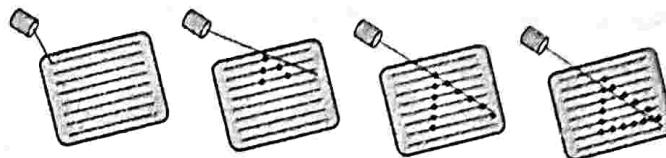


Fig. 1.6.2 : Scene rendering using a raster scan system

- After finishing each scan line (row), electron guns are turned off and moved back to the first pixel of the next scan line. That horizontal movement is called **horizontal retrace**. After reaching to the last pixel of the screen, electron guns are moved at the first pixel of the first row. That vertical movement of electron guns is called **vertical retrace**.

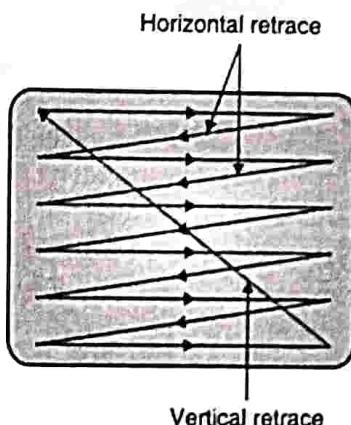


Fig. 1.6.3 : Horizontal and vertical retrace

- The downside of the raster system is that it scans the entire screen even if we want to display a few pixels, as shown in Fig. 1.6.2. It checks for each location in the frame buffer to turn *on* or *off* the electron guns. Pixel is the smallest unit that can be addressed and displayed. Raster scan displays are well suited for realistic scene due to its capability of storing intensity for each pixel. Intensity range for pixel depends on the capability of the raster system.
- A system which supports 1 bit per pixel can display only black and white scene, called a *bi-level system*. A bit value of 1 indicates that the electron beam should be *on* for that location, and *off* otherwise. With n bits per pixel, the system can show maximum 2^n different colors/intensities. Current color CRT system provides 24 bits per pixel, which supports 2^{24} (approximately 17 million) colors.
- Suppose a system has resolution 1280×1280 and it supports 24 bits per pixel. The memory requirement for the frame buffer is,

$$\text{Number of pixels} = 1280 \times 1280 = 16,38,400$$

$$\begin{aligned}\text{Number of bits} &= \text{Number of pixels} \times \text{Number of bits/pixel} \\ &= 16,38,400 \times 24\end{aligned}$$

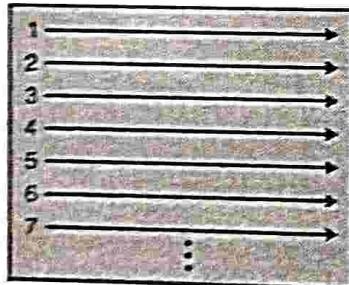
$$\begin{aligned}
 &= 3,93,21,600 \text{ bits} \\
 &= 39321600/8 = 49,15,200 \text{ Bytes} \\
 &= 49,15,200/1024 = 4800 \text{ KB} \\
 &= 4800/1024 = 4.69 \text{ MB}
 \end{aligned}$$

(∴ 1 Byte = 8 bits)
 (∴ 1 KB = 1024 Bytes)
 (∴ 1 MB = 1024 KB)

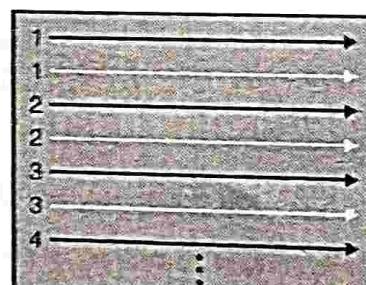
- Thus, the system having screen resolution 1280×1280 and supporting 24 bits per pixel require 4.69 MB of memory for the frame buffer.
- The frame buffer is known as **bitmap** if it uses one bit per pixel. And it is known as **pixmap** if it uses multiple bits per pixel. A bitmap is used in the bi-level system while pixmap is used in a system supporting multiple colors.
- In raster display, the screen is refreshed nearly 60 times per second to display picture without flickering. Rate of the refreshing screen in one second is called **refresh rate**. The refresh rate is measured in frames per second or Hertz. If the refresh rate is set to less than 60, the screen starts flickering. It may happen that while the bottom part of the screen is being painted, the upper part may wipe out due to a lower refresh rate.

Interlaced Display System

- On some raster-scan systems and in TV sets, each frame is displayed in two passes using an **interlaced refresh** procedure. The scene is displayed using half-frames, called **fields**. What we discussed in the previous section is called **progressive display**, in which system display all scan lines one by one. In the interlaced display, all odd scan lines are displayed first, followed by all even scan lines.



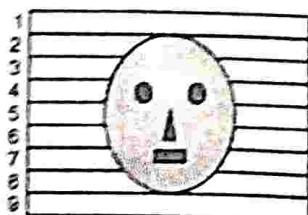
Progressive scan line display



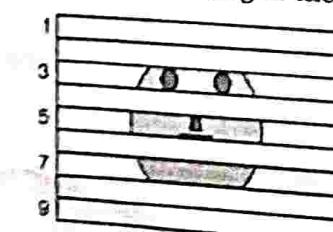
Interlaced scan line display

Fig. 1.6.4 : Displaying using progressive and interlaced scan line

- Each frame contains exactly two fields. The **odd field** is made up of the odd horizontal scan lines in a frame. The odd field is also called the **top field** since it contains the top line of the image. After the vertical retrace, the beam paints out the entire remaining even scan lines called **Even field**. Even the field is called the **bottom field**. Fig. 1.6.5(a) and 1.6.5(b) shows the working of the progressive and interlaced display.



(a) Progressive display



(b) Interlaced display

Fig. 1.6.5 : Progressive vs interlaced display



- Interlaced display overcomes the limitation of the device having slower refreshing rates. On an older system with a refresh rate of 30 Hz, some flicker was noticeable. But by employing the interlaced technique, logically we can double the refresh rate by accomplishing each pass in 1/60 second. This approach makes an effective refresh rate nearer to 60 Hz. This is an effective and efficient way of avoiding flicker.

1.6.3 Random Scan Display

- In a random scan system, the electron beam directly follows the part of the screen where the picture is to be displayed. Instead of an individual pixel, the picture is stored in terms of line drawing commands, and for this reason, it is also referred to as **vector displays** or **stroke-writing** or **calligraphic displays**.
- The memory area where picture definition is stored is referred to as a **refresh display file**. Sometimes it is also called **display list**, **display program**, or **refresh buffer**. In random scan displays, display processor reads line drawing command one by one from the display list and draws it. Fig. 1.6.6 shows the rendering of a polygon on the screen. Electron beam only follows the border of polygon instead of scanning each scan line and deciding which pixel should be turned on or off.



Fig. 1.6.6 : Scene rendering using random scan system

- A pen plotter is the example of a random-scan device. The refresh rate on a random-scan system depends on the number of lines to be displayed. High-quality vector systems are capable of handling near about 10^6 small lines at the refresh rate of 60 Hz. When frame buffer contains a small set of lines, each refresh cycle is delayed to avoid refresh rates greater than 60 Hz. Otherwise, faster refreshing could burn out the phosphor.
- Random-scan systems are useful for line drawing applications and are not suitable to display realistic shaded scenes. Random displays produce smooth line because the CRT beam directly follows the line path, while raster system produces jagged lines. Random scan devices have generally limited color capability.

Sr. No.	Advantages of Random Scan	Disadvantages of Random Scan
1.	Good choice for a device with very high resolution.	Cannot handle the complex or natural scene.
2.	Requires less memory.	Supports limited color only.
3.	Produces smooth lines.	Such devices are costly.
4.	Useful in displaying the static drawing.	Not useful to show animation.

1.6.4 Raster Scan vs. Random Scan

Q. Compare Raster and Random Scan Techniques.

MU - May 18, Dec. 18, 5 Marks

Sr. No.	Raster Scan System	Random Scan System
1.	The electron beam scans the entire screen to draw a picture.	The electron beam scans only the part of the screen where picture information is present.
2.	Due to the discrete nature of the system, it has low resolution.	Electron beam directly follows the path of lines, it draws smooth lines, which gives better resolution.



Sr. No.	Raster Scan System	Random Scan System
3.	Picture definition is stored as a set of discrete intensity values in the frame buffer.	Picture definition is stored as a line drawing commands in the display list.
4.	The intensity value is stored for each pixel, it is well suited to display realistic scene.	The system is designed to display lines, so it cannot be opted to display a realistic scene.
5.	Pixel / spatial location of the screen is used to draw an image.	Mathematical functions are used to draw an image.
6.	The refresh rate is independent of the number of objects in the scene.	When the numbers of primitives are too large, random scan device flickers.
7.	Scan conversion is required.	Scan conversion is not required.
8.	Scan conversion hardware is required as it displays the real-time dynamic scene with extensive computation.	Scan conversion hardware is not required.
9.	Such displays are economical.	They are more costly.
10.	Used to display a dynamic scene.	Used to display static picture.
11.	The video controller is required.	The video controller is not required.

1.6.5 Color CRT Monitors

As we discussed in the previous section, colors are generated by hitting the phosphor layers. Numbers of electrons and speed of electron beam decides the intensity and color of an illuminated pixel. There are two methods to produce color on the screen.

1.6.5.1 Beam Penetration Method

- This method is suitable for random scan system. This method uses two layers of phosphor, red and green. Color of a pixel depends on how far electron beam penetrates these phosphor layers. If an electron beam is fired with slow speed, it only penetrates the outer red phosphor layer.
- And if an electron beam is fired with high speed, it penetrates both layers and excites inner green layer. Intermediate speed produces a mixture of red and green i.e. orange and yellow. Speed of electrons is controlled by beam acceleration voltage.

Advantage

An inexpensive way to produce colors.

Disadvantage

- Suitable to random scan display only.
- Can produce only four colors.

1.6.5.2 Shadow Mask Method

- Beam penetration method cannot render realistic scene properly. Their application is limited to random scan devices only. Another method called shadow mask is used with raster scan devices to produce realistic scene.

- Shadow mask method provides a wider range of colors and intensities. Shadow mask CRT contains three phosphor layers, red, green and blue, each produces a dot at any pixel location. Such CRT has three electron guns, one for each color dot.
- Fig. 1.6.7 shows the delta-delta shadow-mask method. Shadow mask is a mask of holes placed between electron guns and phosphor layers. When the three beams pass through a hole in the shadow mask, they generate a triangle of three dots, which appears as a small color spot on the screen. And this is why such shadow masks are also referred to as *delta shadow mask*.

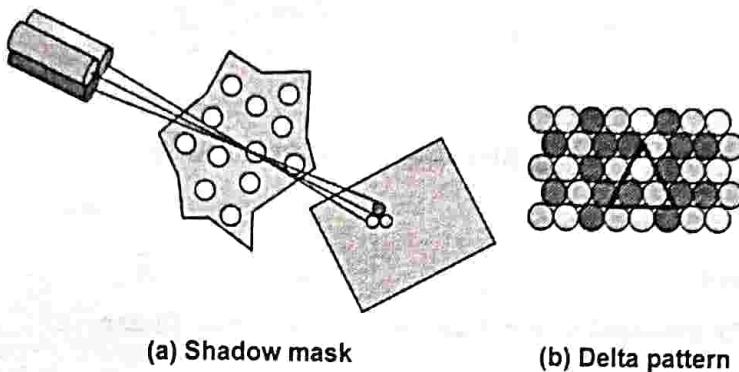


Fig. 1.6.7 : Shadow mask and its operation

- In the *in-line* shadow mask arrangement, three electron guns are aligned and placed side by side. This technique provides higher precision compare to delta shadow mask and should be used in a high resolution system. Color perceived by human eye depends on spatial integration of intensities. So we perceive the average color produced by three electron guns at any given pixel location.
- Numbers of colors on monitor depend on the number of bits allocated per pixel. With n bits per pixel, the system can support 2^n different colors. By varying the voltage levels of electron guns, we can produce different shades of colors. By turning *on* only green and red electron gun, we can produce a yellow color. Similarly magenta is the mixture of red and blue, and cyan can be generated by activating green and blue electron guns.
- In cheaper systems, the electron beam can only be turned to *on* or *off*, which limits the display to only eight colors. Whereas some sophisticated graphics systems allow several million different colors. High-quality raster-graphics systems support 24 bits per pixel, eight-bit per each color, allowing 256 shades of each color. Such a system can produce $256 \times 256 \times 256$ (i.e. approximately 17million) colors. An RGB color system with 24 bits of storage per pixel is generally referred to as a *full-color system* or a *true-color system*.

1.6.6 Direct-View Storage Tubes

Direct View Storage Tube (DVST) is an alternative way to display picture. DVST devices store the picture information inside the CRT itself instead of refreshing the screen. It stores the picture information as a charge distribution just behind the phosphor-coated screen. DVST uses two electron guns, the primary gun is used to store the picture pattern; and the flood gun maintains the picture display.

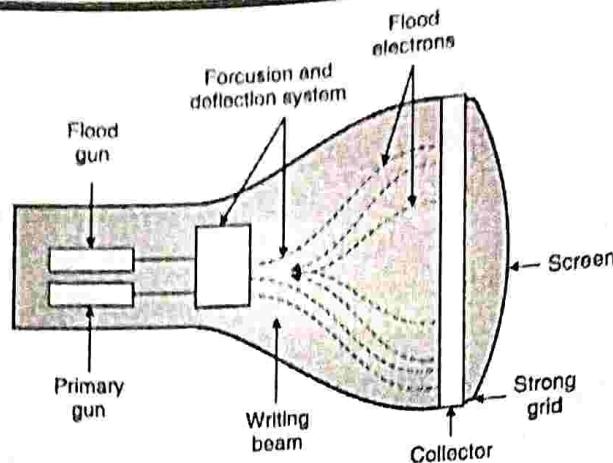


Fig. 1.6.8 : Direct view storage tube

Advantage

- No refreshing is required.
- Complex pictures can be displayed at very high resolutions without flicker.
- Time Consumption is less
- Less expensive

Disadvantage

- Do not display color.
- Selected parts of the picture cannot be erased.
- To manipulate a small part, the entire scene needs to be erased.
- Not useful to display animation.

1.6.7 Flat Panel Display

- Most of the graphics monitors are built using CRTs. The downside of CRT is its size and power consumption. Due to the horizontal arrangement of electron guns, CRT monitors are very thick. Flat-panel displays are thin and lightweight. Even they require less power compare to CRT. We can hang large devices on the wall and small devices are wearable.
- Tablets, digital pads etc. allow users to write on it. Nowadays, flat panel is the first choice for laptop, television set, advertisement boards, displays at commercial blocks etc.
- Flat-panel displays can be classified into two categories, emissive display and non-emissive display :

1. Emissive display devices : They convert electrical energy into the light to generate a display, for example, LED (Light Emitting Diodes), plasma panels, electroluminescent displays etc. Flat CRT is an improved version of CRT, in which electron beams are accelerated parallel to screen rather than perpendicular. Then they are deflected 90 degrees. Although such flat CRT reduces the volume, they are not as successful as other emissive display devices.

2. Non-emissive display devices : They use the optical effect to convert natural or synthetic light to display the graphics on the screen. Example of such a device is LCD (Liquid Crystal Display).

Let us see the working principle of a few flat-panel devices.

1.6.7.1 LCD

- Liquid Crystal Display (LCD) is used with small devices like laptops, calculators, television sets. LCD is a non-emissive display device.
- It uses the concept of polarization to generate a picture. A polarizer is a material which allows passing the light from it having a certain direction only. Light with remaining angle is blocked.
- Grid of crystalline cells called **nematic cells** are used to design LCD. Two light plates, with a polarizer, are arranged at a right angle, opposite to each other. Horizontal conductors and vertical conductors are placed on each plate respectively. The intersection of both plates defines the screen resolution and hence pixel location.

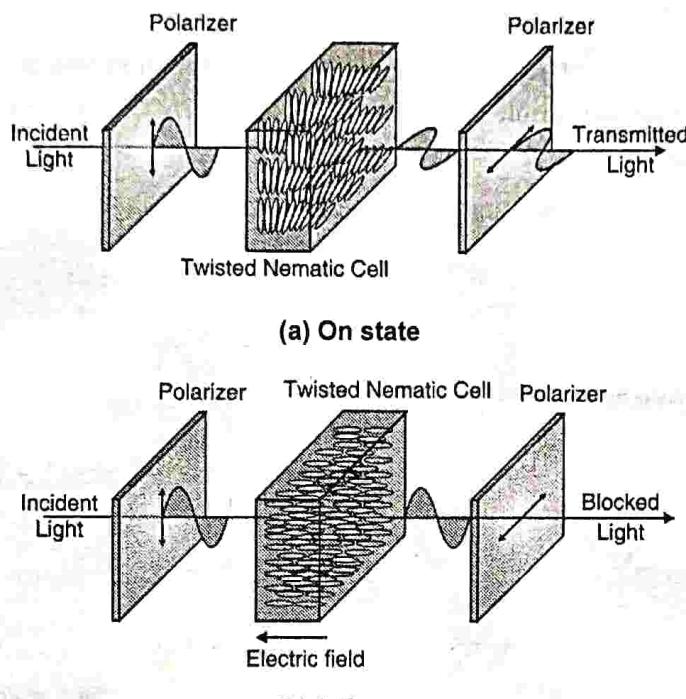


Fig. 1.6.9 : Working of liquid crystal display

- In '*on*' state, nematic cells are arranged as shown in Fig. 1.6.9(a).
- Cell twists the light so that it can pass through the opposite polarizer. The reflected light reaches the eye of the viewer and the scene would be visible. In the *off state*, cells are aligned as shown in Fig. 1.6.9(b). In the *off state*, a cell does not twist the light and so it cannot pass through the opposite polarizer.
- Such a flat-panel device is called **passive matrix LCD**. Another way is to use a transistor at each pixel location is TFT (Thin Film Transistor) technology. Transistors control voltage to decay the charge out of the nematic cell. Such an arrangement is called an **active matrix LCD**.

1.6.7.2 LED

- Light Emitting Diode (LED) display is a matrix of diodes arranged to form the pixel positions in the display. Picture definition is read out from refresh buffer and LEDs on appropriate locations are lit up.
- The LCD gives the best vision when viewing direction is perpendicular to the screen. In LCD, quality of vision reduces with angle. LED displays can sustain the view of angle up to more than 80 degrees and that is why LED is the first choice for advertisement hoardings and monitors.

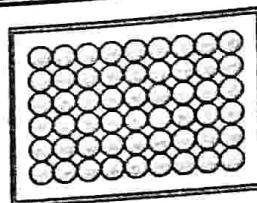


Fig. 1.6.10 : Arrangement of the LED grid

1.6.7.3 Plasma Panels

- Plasma panels are made up of glass plates filled with a mixture of gases like neon. They are also called gas discharge displays due to their characteristic component. Numbers of vertical and horizontal conducting plates are placed on opposite glasses.
- When voltage is applied to conducting plates, gas at the intersection of two conductors glows up the plasma. The device reads the picture definition from a refresh buffer. Pixels are separated by providing an electric field to the conductor plates.

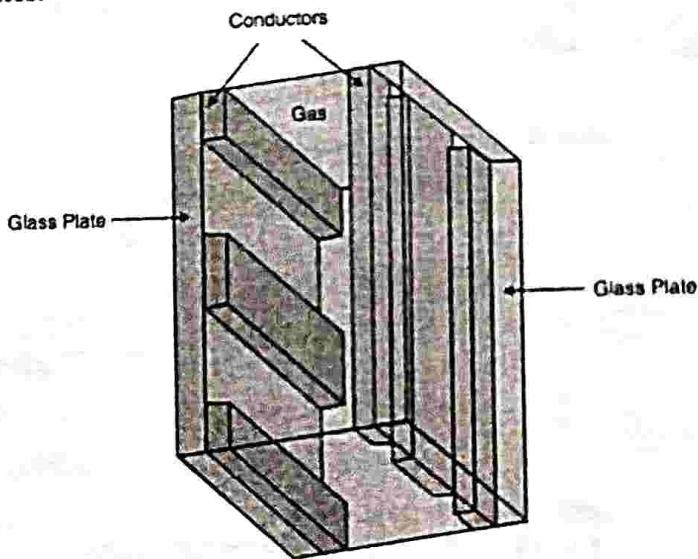


Fig. 1.6.11 : Plasma panel

Disadvantage

Strictly monochromatic.

1.6.7.4 Electroluminescent Displays

- Component arrangement of such display is similar to the plasma panel. The main difference is that the hollow region between opposite glass plates is filled with the phosphor, such as zinc sulfide doped with manganese, instead of gas in the electroluminescent display.
- On applying a high voltage to a pair of crossing electrodes, the phosphor works as a conductor in the intersection region of two electrodes. Generated electrical energy is absorbed by the manganese impurities, which release the energy to become stable and that energy is seen as a light spot similar to the glowing plasma in a plasma panel.

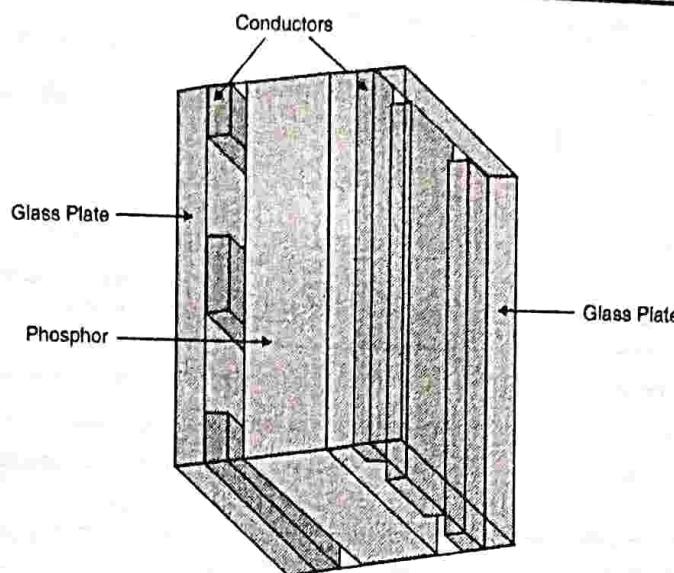


Fig. 1.6.12 : Electroluminescent display

Disadvantage

- More power consumption
- Difficult to achieve good colors or grayscales

1.6.8 Comparison of CRT, LCD, Plasma, and OLED Displays

Sr. No	Parameters for comparison	CRT	LCD	Plasma	OLED
1	Color depth	resolution 4-bit per pixel; It offers better resolution for grayscale	6 to 10-bit per subpixel panels smaller dot pitch, better detail	6 to 8-bit per subpixel panels	8 to 10-bit per subpixel, with some HDR models capable of 12-bit per subpixel.
2	Frame rate	60-85 fps typically, some CRTs can go even higher (200 fps at reduced resolution; internally, display refreshed at input frame rate speed	60 fps typically, gaming monitors can do up to 240 fps; internally, display refreshed at up to 240 Hz	60 fps typically, some can do 120 fps; internally, display refreshed at e.g. 480 or 600 Hz	60 fps typically.
3	Flicker	Noticeable on lower refresh rates (60 Hz and below)	Depends; as of 2013, most LCDs use PWM (strobining) to dim the backlight	Does not normally occur due to high refresh rate.	Does not normally occur.

Sr. No	Parameters for comparison	CRT	LCD	Plasma	OLED
4	Energy consumption and heat generation	High	Low	Varies with brightness but usually higher than LCD	Energy consumption and heat generation
5	Electro-magnetic radiation emission	Can emit a small amount of X-ray radiation.	Only emits non-ionizing radiation.	Emits strong radio frequency electromagnetic radiation	Electro-magnetic radiation emission
6	Environmental influences	Sensitive to ambient magnetic fields, which can adversely affect convergence and color purity.	Prone to malfunctions on both low (below -4 °F/-20 °C) or high (above 45 °C/113 °F) temperatures	High altitude pressure difference may cause poor function or buzzing noises	Can have poor brightness.

1.7 Architecture of Raster Scan Systems

- Hardware architecture of the raster system is shown in Fig. 1.7.1. In addition to the general-purpose CPU, a special-purpose processor, called the **video controller** or **display controller**, is used to control the operation of the display device.

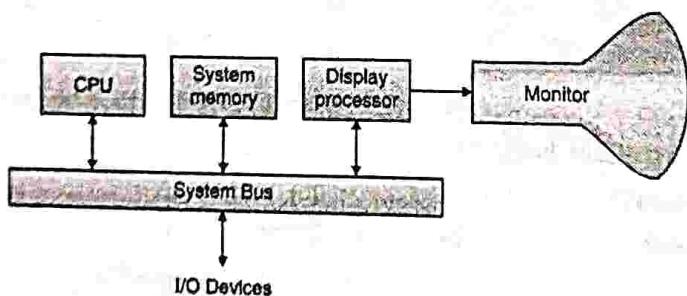


Fig. 1.7.1 : Hardware architecture of the raster graphics system

- In this architecture, the frame buffer is not allocated dedicated memory. System memory and frame buffer share the common memory area. Frame buffer can be anywhere in the system memory. Video controller reads the command from the frame buffer and draws pixels accordingly. Some sophisticated raster systems employ coprocessors and accelerators in addition to the video controller to boost the performance.

1.7.1 Video Controller

Fig. 1.7.2 shows another variation of the raster scan hardware organization. The dedicated memory area is allocated to the frame buffer. The video controller is given direct access to the frame buffer. This organization can render the scene quickly because video controller does not have to put a request on the system bus to read intensity values from the frame buffer.

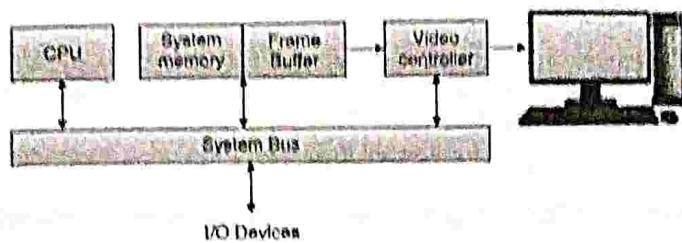


Fig. 1.7.2 : Architecture of a raster system with a reserved memory area for frame buffer

- In some system, the origin is set at the bottom left location of the screen. X coordinate increases from left to right and Y coordinate increase from bottom to top. In some system, the origin is set at top left corner, Y coordinate increases on moving from top to bottom. The scan line is accessed by specifying the Y coordinate.
- Fig. 1.7.3 shows the operation of the video controller. Two registers are used to store the pixel location (x, y). The initial value of x and y are set to 0 and y_{\max} respectively.
- Intensity value from the frame buffer is retrieved from the current (x, y) location. After painting each pixel, x register is incremented by 1 and the process is repeated until x value hits to X_{\max} . After processing all pixel on the current scan line, the content of x register is set to zero and y is decremented by 1 and video controller process the new scan line in the same way. On reaching to $(x_{\max}, 0)$, the controller resets the value of register pair to $(0, y_{\max})$ and repeat the procedure.
- The typical refresh rate is 60 frames per second. In this discussion, we assume that the origin is at the top left corner of the screen.

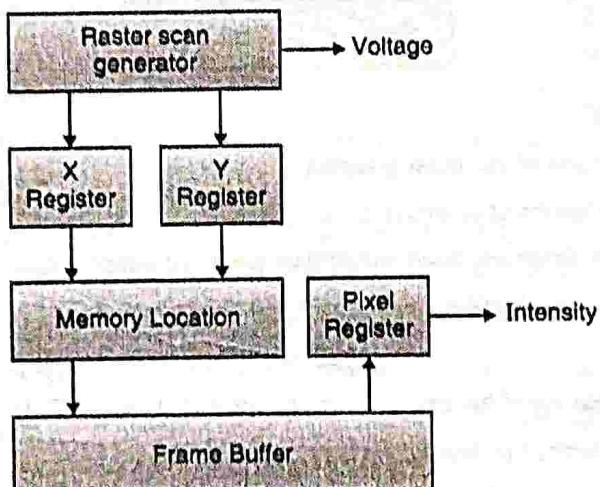


Fig. 1.7.3 : Basic video-controller refresh operations

- High-quality display system uses two frame buffers, so that one buffer can be used for refreshing while other is being filled. Then both buffers switch the role. This mechanism is useful to achieve real-time animation because it does not waste time in reloading the buffer.
- Sometimes the video controller contains a lookup table. Lookup table provides a fast mechanism for changing pixel intensity.
- Pixel value itself in the frame buffer is used to access the intensity value from the lookup table. We do not need to control CRT beam intensity if the lookup table method is used.

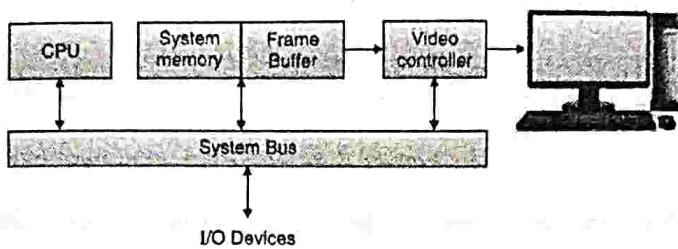


Fig. 1.7.2 : Architecture of a raster system with a reserved memory area for frame buffer

- In some system, the origin is set at the bottom left location of the screen. X coordinate increases from left to right and Y coordinate increase from bottom to top. In some system, the origin is set at top left corner, Y coordinate increases on moving from top to bottom. The scan line is accessed by specifying the Y coordinate.
- Fig. 1.7.3 shows the operation of the video controller. Two registers are used to store the pixel location (x, y). The initial value of x and y are set to 0 and y_{\max} respectively.
- Intensity value from the frame buffer is retrieved from the current (x, y) location. After painting each pixel, x register is incremented by 1 and the process is repeated until x value hits to X_{\max} . After processing all pixel on the current scan line, the content of x register is set to zero and y is decremented by 1 and video controller process the new scan line in the same way. On reaching to $(x_{\max}, 0)$, the controller resets the value of register pair to $(0, y_{\max})$ and repeat the procedure.
- The typical refresh rate is 60 frames per second. In this discussion, we assume that the origin is at the top left corner of the screen.

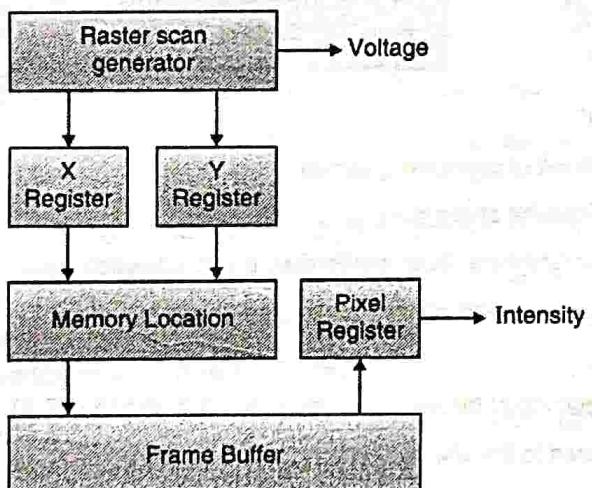


Fig. 1.7.3 : Basic video-controller refresh operations

- High-quality display system uses two frame buffers, so that one buffer can be used for refreshing while other is being filled. Then both buffers switch the role. This mechanism is useful to achieve real-time animation because it does not waste time in reloading the buffer.
- Sometimes the video controller contains a lookup table. Lookup table provides a fast mechanism for changing pixel intensity.
- Pixel value itself in the frame buffer is used to access the intensity value from the lookup table. We do not need to control CRT beam intensity if the lookup table method is used.



1.8 Architecture of Random Scan Systems

- Arrangement of display processor and memory unit for random scan system is shown in Fig. 1.8.1. Application program resides in system memory. Graphics package translates the graphics commands in the application program into a display file. Display file is also stored in system memory.
- Display processor accesses the content of the display file and renders it on the monitor screen.
- Display processor retrieves one by one command from the display file and draws it on screen. Display processor in a random-scan system is also called graphics controller.

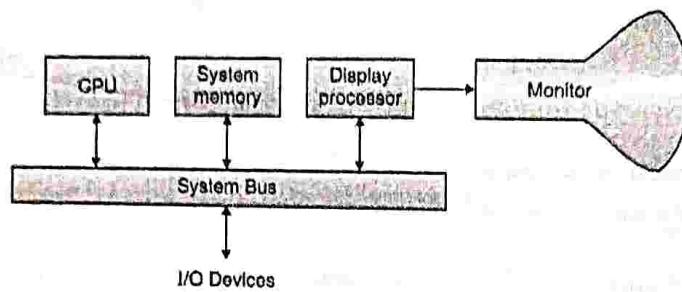


Fig. 1.8.1 : Architecture of a simple random scan system

- Random-scan devices render the scene using short lines. Electron beam tracks the line position directly, rather than going through each pixel on the screen.
- The entire scene is defined using segments of the line in the application program. Line segments are defined by a pair of endpoints. Voltage is adjusted to deflect the electron beam across the line segment.

Review Questions

- Q. 1 What is computer graphics?
- Q. 2 Enlist the classes of operations of computer graphics
- Q. 3 State and explain the applications of computer graphics.
- Q. 4 Define the terms: Computer Graphics, Scan conversion, pixel, resolution, aspect ratio, frame buffer
- Q. 5 Why the border of the object displayed on the monitor screen is zappy ?
- Q. 6 Write short note on vector.
- Q. 7 Find the direction and magnitude of the vector joining points A(2, 8) and B(7, 4).
- Q. 8 Explain how pixels are related to the display memory.
- Q. 9 Find the size of the frame buffer for the system supporting binary colors and having resolution 680 x 420.
- Q. 10 Explain the concept of odd and even field in an interlaced display.
- Q. 11 State the advantages and disadvantages of random scan display.
- Q. 12 Explain the way of computing pixel positions using the slope-Intercept formula.
- Q. 13 Explain the explicit formula of line drawing.



Module 2

Syllabus

Scan conversions of point, line, circle and ellipse : DDA algorithm and Bresenham algorithm for line drawing, midpoint algorithm for circle, midpoint algorithm for ellipse drawing (Mathematical derivation for above algorithms is expected)

Aliasing, Antialiasing techniques like Pre and post filtering, super sampling, and pixel phasing).

Output Primitives

2.1 Scan Conversion

2.1.1 What is Scan Conversion?

- Scan conversion is the process of representing continuous graphics as a collection of discrete pixels. Each pixel on the monitor screen is either turned *on* or *off* according to the picture definition stored in a frame buffer.
- Scan conversion is widely used in CRT, Flat-panel, TV set and other types of display devices to display.
- Implementation of such algorithm varies from one computer system to others. Some algorithms are implemented in software, some are implemented in firmware, and some are the combination of hardware, software and firmware.
- Algorithms implemented in hardware scan converts the objects quicker than software.
- Application program takes these parameters as input and generates all the pixels representing the shape. This process is called scan conversion.
- All the points on a line can be generated using DDA or Bresenham's algorithm for given endpoints.
- Most of the monitors and television sets support raster scan display. In raster scan devices, picture definition is stored as a discrete pixel value. The entire scene is defined by a set of intensity values for each pixel on the screen.
- Computed pixel values may be a fraction, so it needs to round off to show it on a monitor screen.
- As shown in Fig. 2.1.1, computed pixel values (10.8, 10.8), (12.6, 11.7) and (13.3, 10.4) are rounded to nearest integer values (11, 11), (13, 12) and (13, 10).

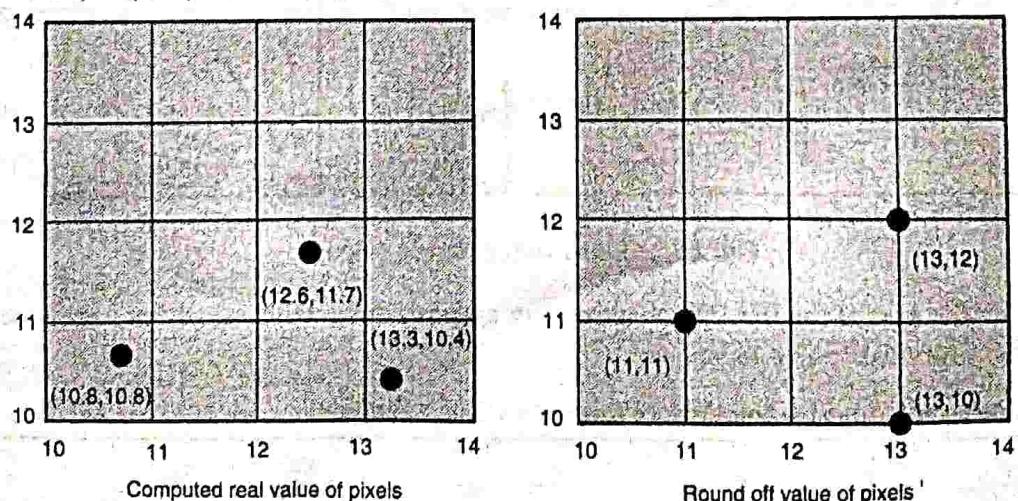


Fig. 2.1.1 : Conversion from actual value to rounded value

2.1.2 Line and Line Segment

- The line is a collection of points along a straight path. Typically, the line is described by its two end points (x_1, y_1) and (x_2, y_2) . Intermediate points on the line are computed by the line generation algorithm.
- Geometric interpretation of line joining end points P_1 and P_2 are depicted in Fig. 2.1.2

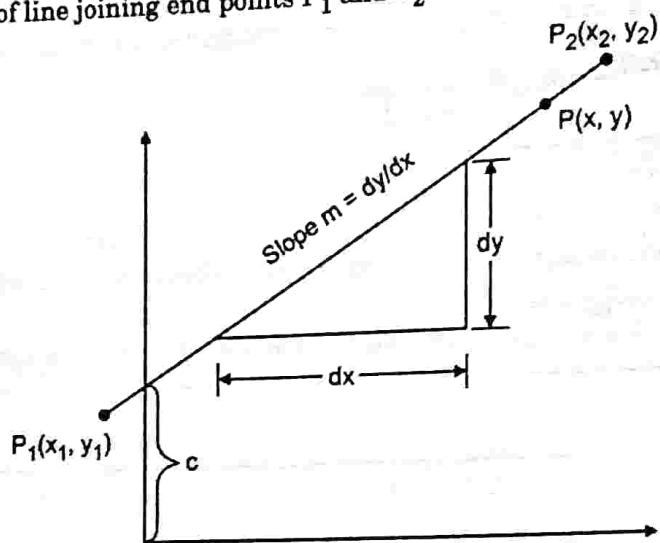


Fig. 2.1.2 : Representation of line

- Given the end points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, any point $P(x, y)$ on the line can be computed using the slope-intercept formula as follow.
- It is evident from Fig. 2.1.2 that the slope of the line is,

$$m = \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1}$$

- Let 'c' represents the Y-intercept of the line. Given the x-coordinate, the value of the corresponding y-coordinate of any point is computed as follow :

$$y = mx + c$$

- A horizontal line has slope 0. The line bisecting the first quadrant has slope 1. As the angle of line with X-axis increases, the slope increases. The slope of the vertical line is ∞ .
- Lines with different slopes are depicted in Fig. 2.1.3.

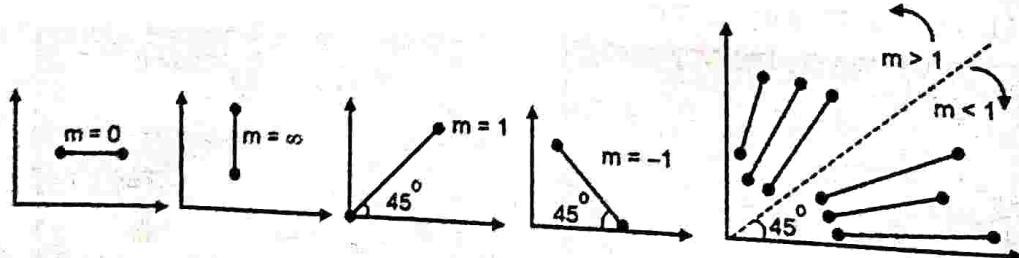
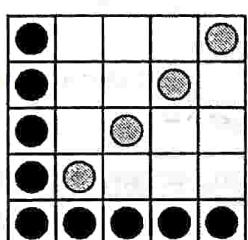


Fig. 2.1.3 : Lines with different slope

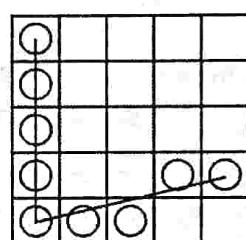
- A line segment is the part of the line. Often curves are approximated using small line segments. Almost any shape can be approximated by line segments.
- Like line, the line segment is also bounded by some start and endpoints.

2.1.3 Qualities of Good Line Drawing Algorithms

- When we draw a line on paper, it looks smooth and continuous. But due to the grid structure of monitor, when the line is rasterized, it looks zig-zag except horizontal ($m = 0$), vertical ($m = \infty$) and diagonal ($m = 1$) line.
 - Distance between pixels also changes with orientation, and hence the brightness.
- An ideal line should have the following properties :
- It should interpolate both the endpoints.
 - The brightness of the line should be independent of the orientation of the line.
 - It should appear straight and smooth.
 - It should be drawn quickly.
 - All properties are satisfied when we draw lines on paper. But when we rasterize it on the monitor, most of the properties do not hold.
 - On the monitor screen, vertical and horizontal lines are displayed with maximum intensity; the brightness of inclined line changes with orientation.



Line with different Intensities



Line with different orientations

Fig. 2.1.4 : Line with different Intensities and orientations

- All properties are satisfied when we draw them on paper. But when we rasterize it on the monitor, most of the properties do not hold.
- On the monitor screen, vertical and horizontal lines are displayed with maximum intensity; the brightness of inclined line changes with orientation

2.2 Line Drawing Algorithms

Line generation or scan conversion of the line is the process of turning 'on' or 'off' the pixel along the line path.

In this section, we will discuss the following line drawing algorithms.

1. Digital differential analyzer
2. Bresenham's Line Drawing Algorithm

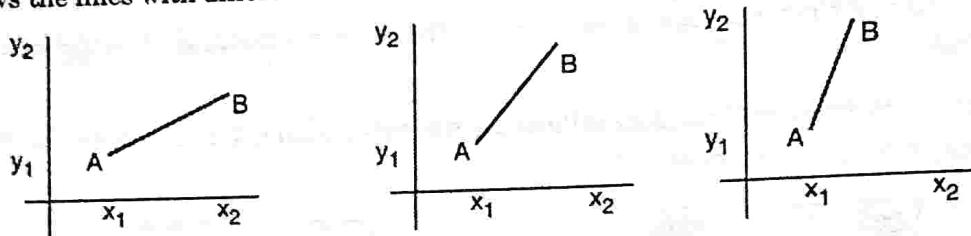
2.2.1 DDA Line Drawing Algorithm

Q. What are the disadvantages of DDA algorithm?

MU - Dec. 18. 5 Marks

2.2.1.1 Working Mechanism

- Digital Differential Analyzer (DDA) is a simple, incremental line scan converting algorithm.
- In the DDA algorithm, either horizontal or vertical displacement is set to unit interval and the corresponding displacement for other direction is calculated using the slope.
- If the line makes an angle less than 45° with X-axis (i.e. $m < 1$), increment in the X-direction is set to 1 and corresponding Y is computed.
- And if line makes an angle greater than 45° with X-axis (i.e. $m > 1$), increment in the Y-direction is set to 1 and corresponding X is computed.
- For line with slope $m = 1$, increment in both the directions is set to 1.
- Fig. 2.2.1 shows the lines with different orientations.



(a) Line with slope $|m| < 1$ (b) Line with slope $|m| = 1$ (c) Line with slope $|m| > 1$

Fig. 2.2.1 : Lines with different orientations

- Almost all recent monitors support integer coordinate system. Only integer movement is possible in any direction. It is not possible to draw a pixel at position (2.3, 5.6), the system either rounds up or round down coordinate values and then display the point.
- We will derive the equation to draw a line in the *first quadrant*.
- Another assumption is that we will process the line from *left to right*. It is intuitive to derive a line in any other quadrant just by changing the sign of increment values.
- Let's consider that Δx and Δy are the increments in x and y direction, respectively. Value of x and y-coordinates at location $(k + 1)$ is given by,

$$\text{New value} = \text{Old value} + \text{Increment}$$

$$X_{k+1} = x_k + \Delta x$$

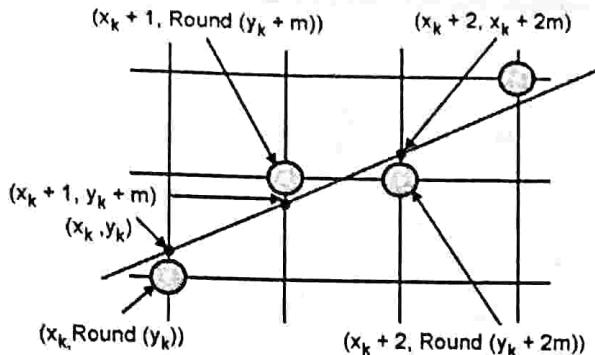
$$Y_{k+1} = y_k + \Delta y$$

We will discuss three cases.

Case - I : Slope is less than 1 ($|m| < 1$)

When the slope is less than 1, Δx is set to the unit interval, i.e., $\Delta x = 1$ and corresponding y-coordinate is computed.

Fig. 2.2.2 shows the actual line and the rounded off locations of a pixel on the monitor grid.

Fig. 2.2.2 : Line with $|m| < 1$

From equation of slope,

$$m = \frac{\Delta y}{\Delta x} = \frac{\Delta y}{1}$$

$$\therefore \Delta y = m$$

So coordinate value for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k + 1$$

$$y_{k+1} = y_k + \Delta y = y_k + m$$

Value of m is real number, so y_{k+1} may be real number. Hence, every time after adding m to y_k , answer is rounded to nearest integer.

If the line is processed from right to left, the value of x and y decreases in every iteration. So, Δx is set to -1 and Δy is set to $-m$.

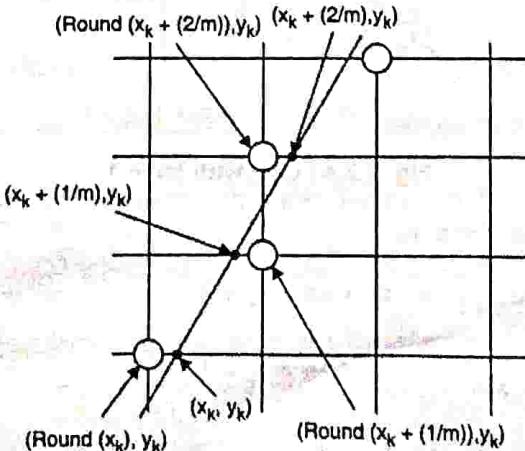
In that case, coordinate values for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k - 1$$

$$y_{k+1} = y_k + \Delta y = y_k - m$$

Case - II : Slope is greater than 1 ($|m| > 1$)

When slope is greater than 1, Δy is set to unit interval, i.e., $\Delta y = 1$ and corresponding x -coordinate is computed.

Fig. 2.2.3 : Line with $|m| > 1$

From equation of slope,

$$m = \frac{\Delta y}{\Delta x} = \frac{1}{m}$$

$$\therefore \Delta x = \frac{1}{m}$$

Coordinate value for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k + \frac{1}{m}$$

$$y_{k+1} = y_k + \Delta y = y_k + 1$$

If the line is processed from right to left, the value of x and y decreases in every iteration. Hence, Δx is set to $(-\frac{1}{m})$ and Δy is set to -1 .

In that case, coordinate values for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k - \frac{1}{m}$$

$$y_{k+1} = y_k + \Delta y = y_k - 1$$

Case - III : Slope is 1 ($|m| = 1$)

When slope of line is 1, i.e. line makes an angle of 45° with X-axis, we increment both coordinate value by 1. So, $\Delta x = \Delta y = 1$.

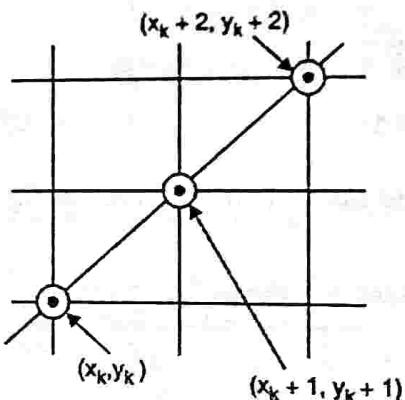


Fig. 2.2.4 : Line with $|m| = 1$

Coordinate value for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k + 1$$

$$y_{k+1} = y_k + \Delta y = y_k + 1$$

If the line is processed from right to left, in that case, the value of x and y decreases in every iteration. Δx and Δy are set to -1 .



In that case coordinate values for next pixel is given as,

$$x_{k+1} = x_k + \Delta x = x_k - 1$$

$$y_{k+1} = y_k + \Delta y = y_k - 1$$

Table 2.2.1 shows the sign and value of x and y increments to draw a line in any quadrant.

Table 2.2.1 : Generalization of DDA line drawing algorithm

Quadrant	Magnitude of Slope	If processed from left to right		If processed from right to left	
		Δx	Δy	Δx	Δy
1	$ m < 1$	1	m	-1	$-m$
	$ m > 1$	$1/m$	1	$-1/m$	-1
2	$ m < 1$	1	$-m$	-1	m
	$ m > 1$	$-1/m$	-1	$1/m$	1
3	$ m < 1$	1	m	-1	$-m$
	$ m > 1$	$-1/m$	1	$1/m$	-1
4	$ m < 1$	1	$-m$	-1	m
	$ m > 1$	$1/m$	-1	$-1/m$	1

2.2.1.2 Algorithm

Steps of DDA Line Drawing Algorithm :

Step 1 : Read two endpoints (x_1, y_1) and (x_2, y_2)

Step 2 : Compute horizontal and vertical differences as,

$$dx = x_2 - x_1$$

$$dy = y_2 - y_1$$

Step 3 : Set number of steps to $N = \max(|dx|, |dy|)$

Step 4 : Repeat step 5 to step 8 N times.

Step 5 : If $|dx| > |dy|$ and $x_1 < x_2$, set increments in X and Y-direction to 1 and m , respectively.

Step 6 : If $|dx| > |dy|$ and $x_1 \geq x_2$, set increments in X and Y-direction to (-1) and (- m), respectively.

Step 7 : If $|dx| < |dy|$ and $y_1 < y_2$, set increments in X and Y-direction to $1/m$ and 1, respectively.

Step 8 : If $|dx| < |dy|$ and $y_1 \geq y_2$, set increments in X and Y-direction to (- $1/m$) and (-1), respectively.

Note: These steps are to draw a line in the first quadrant – growing from left to right or right to left only. Generalized line algorithm can be derived with the help of increment and sign as shown in Table 2.2.1.

DDA line drawing algorithm

Algorithm to draw a line in first quadrant, growing from left to right is stated as follows :

Algorithm DDA_LINE (x_1, y_1, x_2, y_2)

```

dx ←  $x_2 - x_1$ 
dy ←  $y_2 - y_1$ 
x ←  $x_1$ 
y ←  $y_1$ 
m ← dy/dx
if abs(m) < 1 then
    numSteps ← abs(dx)
else
    numSteps ← abs(dy)
end
Δx ← dx/numSteps
Δy ← dy/numSteps
putPixel (x, y)
for x ←  $x_1$  to  $x_2$ , do
    x ← x + Δx
    y ← y + Δy
    putPixel (Round(x), Round(y))
end

```

This algorithm draws a line from left to right in the first quadrant.

2.2.1.3 Pros and Cons

Advantages and disadvantages of DDA line drawing algorithm are stated here :

Advantages

1. It is simple.
2. It is easy to understand.
3. It eliminates multiplications involved in explicit line drawing equation, $y = mx + c$.
4. DDA is quite faster.
5. DDA is more efficient than an implicit line drawing algorithm.
6. It does not require any special skill to implement it.

Disadvantages

1. It involves floating-point operation for each pixel.
2. It performs rounding off operation for each pixel.
3. Rounding off error is accumulated in each iteration and calculated pixel position may drift away from the actual position due to cumulative rounding off error.
4. It is slower.

2.2.1.4 Examples

Ex. 2.2.1 : Calculate the pixel coordinates of line AB using the DDA algorithm. Where, A = (0, 0) and B = (4, 6).

Soln. :

Let's take, $(x_1, y_1) = (0, 0)$ and $(x_2, y_2) = (4, 6)$

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6 - 0}{4 - 0} = \frac{6}{4} = 1.5$$

Line is in first quadrant, it is growing from left to right and slope $|m| > 1$, so set $\Delta x = 1/m = 0.67$ and $\Delta y = 1$. So,

$$x_{k+1} = x_k + \Delta x = x_k + 0.67$$

$$y_{k+1} = y_k + \Delta y = y_k + 1$$

Table P. 2.2.1 shows the calculation for required pixels.

Table P. 2.2.1 : Calculation for required pixels

k	(x_k, y_k)	$x_{k+1} = x_k + 0.67$	$y_{k+1} = y_k + 1$	Round (x_{k+1})
0	(0, 0)	$0 + 0.67 = 0.67$	$0 + 1 = 1$	1
1	(1, 1)	$0.67 + 0.67 = 1.34$	$1 + 1 = 2$	1
2	(1, 2)	$1.34 + 0.67 = 2.01$	$2 + 1 = 3$	2
3	(2, 3)	$2.01 + 0.67 = 2.68$	$3 + 1 = 4$	3
4	(3, 4)	$2.68 + 0.67 = 3.35$	$4 + 1 = 5$	3
5	(3, 5)	$3.35 + 0.67 = 4.02$	$5 + 1 = 6$	4
6	(4, 6)	-	-	-

In Table P. 2.2.1, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.1.

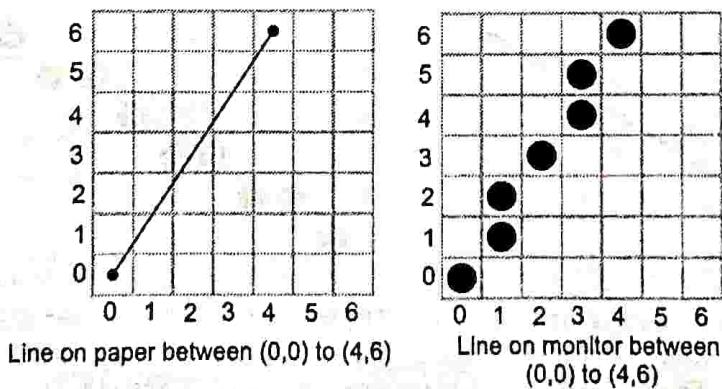


Fig. P. 2.2.1 : DDA line between points (0, 0) and (4, 6)

Ex. 2.2.2 : Draw a line from point (2, 2) to (10, 7) using DDA line drawing algorithm.

Soln. :

Let's take, $(x_1, y_1) = (2, 2)$ and $(x_2, y_2) = (10, 7)$

$$\text{Slope of line } = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7 - 2}{10 - 2} = \frac{5}{8} = 0.625$$

Line is in first quadrant, it is growing from left to right and slope $|m| < 1$, so set $\Delta x = 1$ and $\Delta y = m = 0.625$.
So,

$$x_{k+1} = x_k + \Delta x = x_k + 1$$

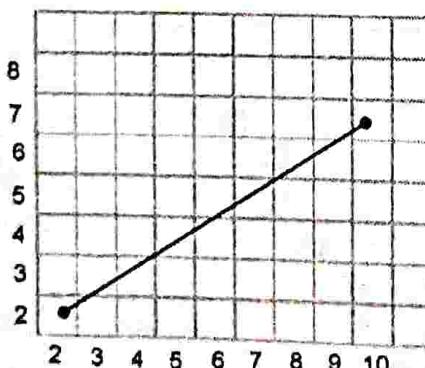
$$y_{k+1} = y_k + \Delta y = y_k + 0.625$$

Table P. 2.2.2 shows the calculation for required pixels.

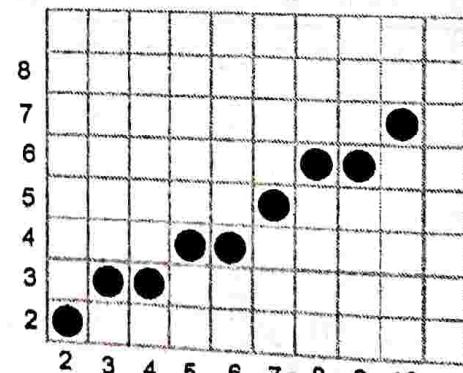
Table P. 2.2.2 : Calculation for required pixels

k	(x _k , y _k)	X _{k+1} = x _k + 1	y _{k+1} = y _k + 0.625	Round (y _{k+1})
0	(2, 2)	2 + 1 = 3	2 + 0.625 = 2.625	3
1	(3, 3)	3 + 1 = 4	2.625 + 0.625 = 3.25	3
2	(4, 3)	4 + 1 = 5	3.25 + 0.625 = 3.875	4
3	(5, 4)	5 + 1 = 6	3.875 + 0.625 = 4.5	5
4	(6, 5)	6 + 1 = 7	4.5 + 0.625 = 5.125	5
5	(7, 5)	7 + 1 = 8	5.125 + 0.625 = 5.75	6
6	(8, 6)	8 + 1 = 9	5.75 + 0.625 = 6.375	6
7	(9, 6)	9 + 1 = 10	6.375 + 0.625 = 7.000	7
8	(10, 7)	-	-	-

In Table P. 2.2.2, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.2.



The line on paper between (2, 2) to (10, 7)



The line on the monitor between (2, 2) to (10, 7)

Fig. P. 2.2.2 : DDA line between points (2, 2) and (10, 7)

Ex. 2.2.3 : Draw a line from A(10, 2) to B(6, 8) using DDA line drawing algorithm.

Soln. :

Let's take, (x₁, y₁) = (10, 2) and (x₂, y₂) = (6, 8)



$$\text{Slope of line } = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8 - 2}{6 - 10} = \frac{6}{-4} = -1.5$$

The line is in the first quadrant, it is growing from right to left and slope $m < 1$. It is intuitive from the line coordinates that when line proceeds from point A to point B, its x-coordinate decreases and y-coordinate increases, so set

$$\Delta y = 1 \text{ and } \Delta x = \frac{1}{m} = -0.67$$

$$x_{k+1} = x_k + \Delta x = x_k - 0.67$$

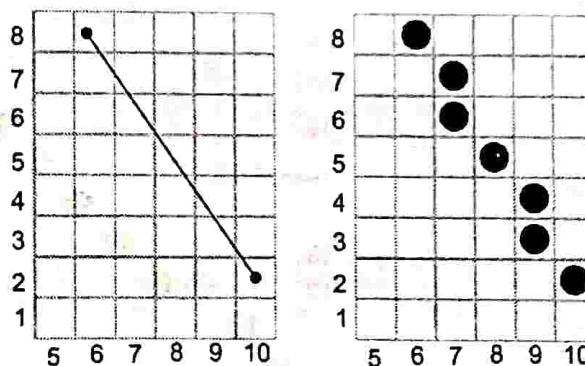
$$y_{k+1} = y_k + \Delta y = y_k + 1$$

Table P. 2.2.3 shows the calculation for required pixels.

Table P. 2.2.3 : Calculation for required pixels

k	(x_k, y_k)	$x_{k+1} = x_k - 0.67$	$y_{k+1} = y_k + 1$	Round (x_{k+1})
0	(10, 2)	$10 - 0.67 = 9.33$	$2 + 1 = 3$	9
1	(9, 3)	$9.33 - 0.67 = 8.67$	$3 + 1 = 4$	9
2	(9, 4)	$8.67 - 0.67 = 8.00$	$4 + 1 = 5$	8
3	(8, 5)	$8.00 - 0.67 = 7.33$	$5 + 1 = 6$	7
4	(7, 6)	$7.33 - 0.67 = 6.67$	$6 + 1 = 7$	7
5	(7, 7)	$6.67 - 0.67 = 6.00$	$7 + 1 = 8$	6
6	(6, 8)	-	-	-

In Table P. 2.2.3, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.3.



The line on paper between
(10,2) to (6,8)

The line on the Monitor between
(10,2) to (6,8)

Fig. P. 2.2.3 : DDA line between points (10, 2) and (6, 8)

Ex. 2.2.4 : Draw a line from A (20, 10) to B (30, 18) using DDA line drawing algorithm.

Soln. :

Let's take $(x_1, y_1) = (20, 10)$ and $(x_2, y_2) = (30, 18)$

$$\text{Slope of line } = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{18 - 10}{30 - 20} = \frac{8}{10} = 0.8$$

Line is in first quadrant, it is growing from left to right and slope $|m| < 1$, so set $\Delta x = 1$ and $\Delta y = m = 0.8$.

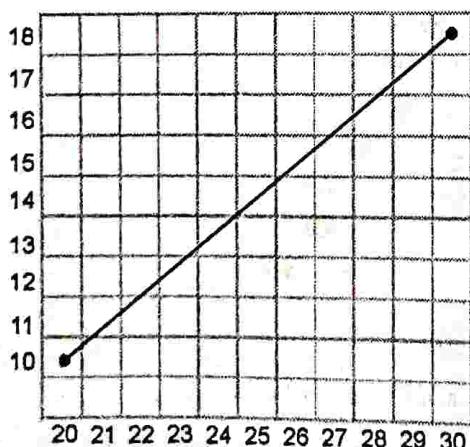
$$X_{k+1} = x_k + \Delta x = x_k + 1$$

$$y_{k+1} = y_k + \Delta y = y_k + 0.8$$

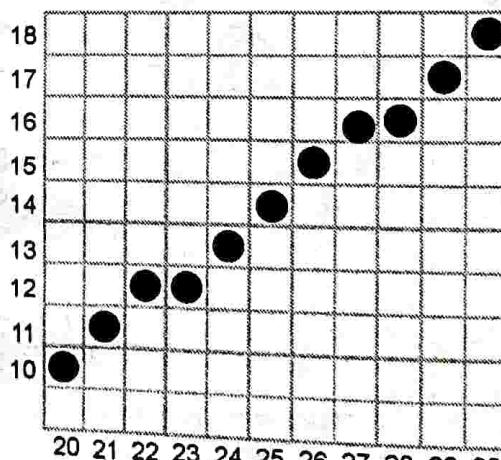
Table P. 2.2.4 : Calculation for required pixels

k	(x_k, y_k)	$X_{k+1} = x_k + 1$	$y_{k+1} = y_k + 0.8$	Round (y_{k+1})
0	(20, 10)	$20 + 1 = 21$	$10.0 + 0.8 = 10.8$	11
1	(21, 11)	$21 + 1 = 22$	$10.8 + 0.8 = 11.6$	12
2	(22, 12)	$22 + 1 = 23$	$11.6 + 0.8 = 12.4$	12
3	(23, 12)	$23 + 1 = 24$	$12.4 + 0.8 = 13.2$	13
4	(24, 13)	$24 + 1 = 25$	$13.2 + 0.8 = 14.0$	14
5	(25, 14)	$25 + 1 = 26$	$14.0 + 0.8 = 14.8$	15
6	(26, 15)	$26 + 1 = 27$	$14.8 + 0.8 = 15.6$	16
7	(27, 16)	$27 + 1 = 28$	$15.6 + 0.8 = 16.4$	16
8	(28, 16)	$28 + 1 = 29$	$16.4 + 0.8 = 17.2$	17
9	(29, 17)	$29 + 1 = 30$	$17.2 + 0.8 = 18.0$	18
10	(30, 18)	-	-	-

In Table P. 2.2.4, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.4.



The line on paper between (20, 10) to (30, 18)



The line on the Monitor
between (20, 10) to (30, 18)

Fig. P. 2.2.4 : DDA line between points (20, 10) and (30, 18)

Ex. 2.2.5 : Draw a line from A (0, 0) to B (-5, -5) using the DDA line drawing algorithm.

Soln. :

Let's take, $(x_1, y_1) = (0, 0)$ and $(x_2, y_2) = (-5, -5)$

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-5 - 0}{-5 - 0} = \frac{-5}{-5} = 1$$

Line is in third quadrant, it is growing from right to left and slope $|m| = 1$, so set $\Delta x = -1$ and $\Delta y = -1$

$$x_{k+1} = x_k + \Delta x = x_k - 1$$

$$y_{k+1} = y_k + \Delta y = y_k - 1$$

Table P. 2.2.5 : Calculation for required pixels

k	(x_k, y_k)	$x_{k+1} = x_k - 1$	$y_{k+1} = y_k - 1$	Round (x_{k+1}, y_{k+1})
0	(0, 0)	$0 - 1 = -1$	$0 - 1 = -1$	(-1, -1)
1	(-1, -1)	$-1 - 1 = -2$	$-1 - 1 = -2$	(-2, -2)
2	(-2, -2)	$-2 - 1 = -3$	$-2 - 1 = -3$	(-3, -3)
3	(-3, -3)	$-3 - 1 = -4$	$-3 - 1 = -4$	(-4, -4)
4	(-4, -4)	$-4 - 1 = -5$	$-4 - 1 = -5$	(-5, -5)
5	(-5, -5)	-	-	-

In Table P. 2.2.5, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.5.

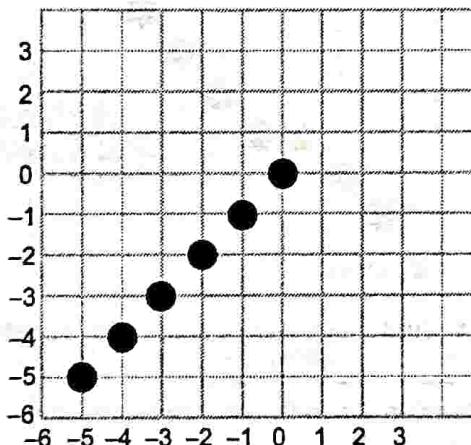


Fig. P. 2.2.5

Ex. 2.2.6 : Scan convert a line with endpoints (10, 5) and (16, 10) using DDA line drawing algorithm.

Soln. :

Let's take, $(x_1, y_1) = (10, 5)$ and $(x_2, y_2) = (16, 10)$

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10 - 5}{16 - 10} = \frac{5}{6} = 0.833$$

Line is in first quadrant, it is growing from left to right and slope $|m| < 1$, so set $\Delta x = 1$ and $\Delta y = m = 0.833$.
So,

$$x_{k+1} = x_k + \Delta x = x_k + 1$$

$$y_{k+1} = y_k + \Delta y = y_k + 0.833$$

Table P. 2.2.6 shows the calculation for the required pixels.

Table P. 2.2.6 : Calculation for required pixels

k	(x_k, y_k)	$x_{k+1} = x_k + 1$	$y_{k+1} = y_k + 0.833$	Round (y_{k+1})
0	(10, 5)	$10 + 1 = 11$	$5 + 0.833 = 5.833$	6
1	(11, 6)	$11 + 1 = 12$	$5.833 + 0.833 = 6.666$	7
2	(12, 7)	$12 + 1 = 13$	$6.666 + 0.833 = 7.499$	7
3	(13, 7)	$13 + 1 = 14$	$7.499 + 0.833 = 8.332$	8
4	(14, 8)	$14 + 1 = 15$	$8.332 + 0.833 = 9.165$	9
5	(15, 9)	$15 + 1 = 16$	$9.165 + 0.833 = 9.998$	10
6	(16, 10)	-	-	-

In Table P. 2.2.6, columns 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.6.

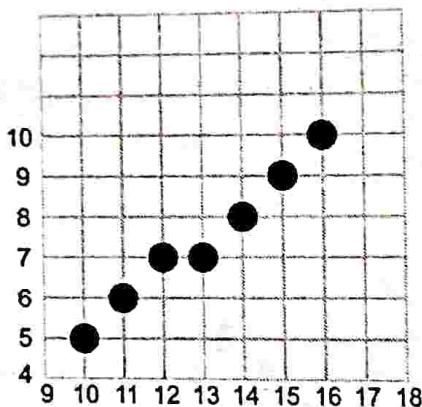


Fig. P. 2.2.6 : DDA line between points (10, 5) and (16, 10)

Ex. 2.2.7 : Explain DDA line drawing algorithm. Consider line segment from A (- 2, - 1) to B (6, 3) use DDA line drawing algorithm to rasterize this line.

Soln. :

Let's take, $(x_1, y_1) = (-2, -1)$ and $(x_2, y_2) = (6, 3)$

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3 - (-1)}{6 - (-2)} = \frac{4}{8} = 0.5$$

Line is in third and first quadrant, it is growing from left to right and slope $|m| < 1$, so set $\Delta x = 1$ and $\Delta y = m = 0.5$. So,

$$x_{k+1} = x_k + \Delta x = x_k + 1$$

$$y_{k+1} = y_k + \Delta y = y_k + 0.5$$

Table P. 2.2.7 shows the calculation for required pixels.

Table P. 2.2.7 : Calculation for required pixels

k	(x _k , y _k)	x _{k+1} = x _k + 1	y _{k+1} = y _k + 0.5	Round (y _{k+1})
0	(-2, -1)	-2 + 1 = -1	-1 + 0.5 = -0.5	0
1	(-1, 0)	-1 + 1 = 0	-0.5 + 0.5 = 0	0
2	(0, 0)	0 + 1 = 1	0 + 0.5 = 0.5	1
3	(1, 1)	1 + 1 = 2	0.5 + 0.5 = 1	1
4	(2, 1)	2 + 1 = 3	1 + 0.5 = 1.5	2
5	(3, 2)	3 + 1 = 4	1.5 + 0.5 = 2	2
6	(4, 2)	4 + 1 = 5	2 + 0.5 = 2.5	3
7	(5, 3)	5 + 1 = 6	2.5 + 0.5 = 3	3
8	(6, 3)	-	-	-

In Table P. 2.2.7, columns 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.7.

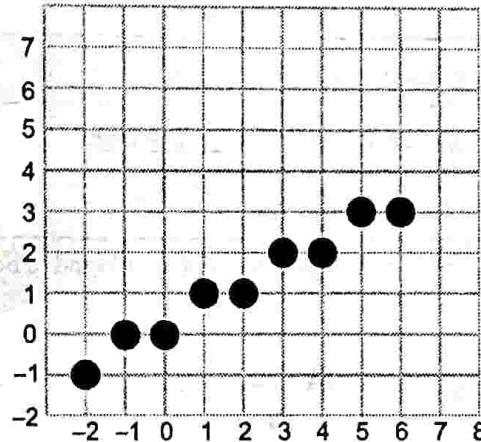


Fig. P. 2.2.7 : DDA line between points (-2, -1) and (6, 3)

Ex. 2.2.8 : Explain DDA Line drawing algorithm and Plot the points for line AB (A (10, 15) B (5, 25) using it.

MU - Dec. 19, 10 Marks

Soln. :

Let's take, (x₁, y₁) = (10, 15) and (x₂, y₂) = (5, 25)

$$\text{Slope of line} = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{25 - 15}{5 - 10} = \frac{10}{-5} = -2.0$$

The line is in the first quadrant, it is growing from right to left and slope m < 1. It is intuitive from the line coordinates that when line proceeds from point A to point B, its x-coordinate decreases and y-coordinate increases, so set

$$\Delta y = 1 \text{ and } \Delta x = \frac{1}{m} = -0.5$$



$$x_{k+1} = x_k + \Delta x = x_k - 0.5$$

$$y_{k+1} = y_k + \Delta y = y_k + 1$$

Table P. 2.2.8 shows the calculation for required pixels.

Table P. 2.2.8 : Calculation for required pixels

k	(x_k, y_k)	$x_{k+1} = x_k - 0.5$	$y_{k+1} = y_k + 1$	Round (x_{k+1})
0	(10, 15)	$10 - 0.5 = 9.5$	$15 + 1 = 16$	10
1	(10, 16)	$9.5 - 0.5 = 9.0$	$16 + 1 = 17$	9
2	(9, 17)	$9.0 - 0.5 = 8.5$	$17 + 1 = 18$	9
3	(9, 18)	$8.5 - 0.5 = 8.0$	$16 + 1 = 19$	8
4	(8, 19)	$8.0 - 0.5 = 7.5$	$19 + 1 = 20$	8
5	(8, 20)	$7.5 - 0.5 = 7.0$	$20 + 1 = 21$	7
6	(7, 21)	$7.0 - 0.5 = 6.5$	$21 + 1 = 22$	7
7	(7, 22)	$6.5 - 0.5 = 6.0$	$22 + 1 = 23$	6
8	(6, 23)	$6.0 - 0.5 = 5.5$	$23 + 1 = 24$	6
9	(6, 24)	$5.5 - 0.5 = 5.0$	$24 + 1 = 25$	5
10	(5, 25)	-	-	-5

In Table P. 2.2.8 , column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.8.

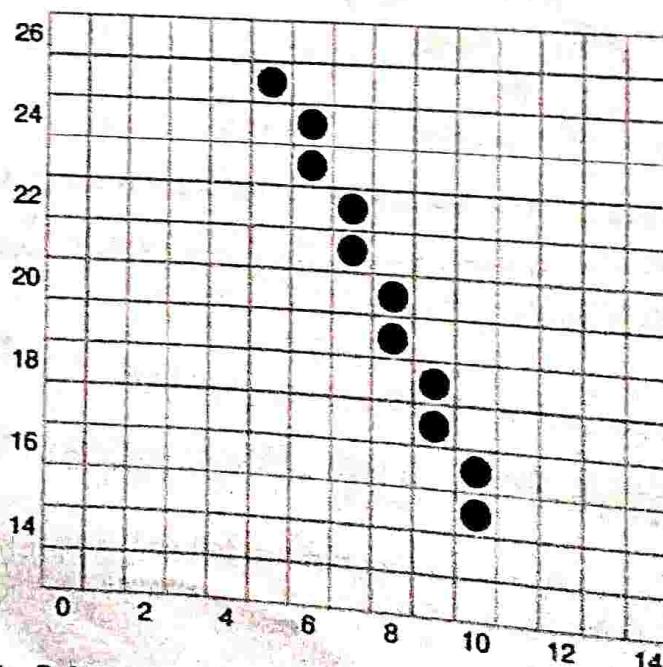


Fig. P. 2.2.8 : DDA line between points (10, 15) and (5, 25)

2.2.2 Bresenham Line Drawing Algorithm

2.2.2.1 Working Mechanism

- DDA algorithm is simple but not efficient. It involves floating-point calculations.
- Bresenham has proposed integer calculation based on an incremental approach to computing the next pixel position along a straight line.
- Given the endpoints of the line (x_0, y_0) and (x_1, y_1) , we will derive the incremental formula for Bresenham's line drawing algorithm.
- This approach selects the pixel which is close to the ideal line. It uses only integer addition and subtraction operations.
- **Assumption :** Line is in the first quadrant, growing from left to right and $|m| < 1$.

The slope of the line is given as, $m = \frac{y_1 - y_0}{x_1 - x_0}$.

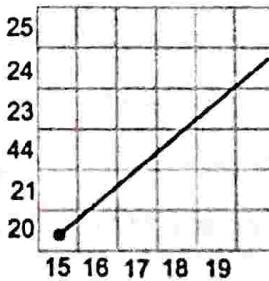


Fig. 2.2.5 : Segment of the line

- Fig. 2.2.5 shows the segment of the line. The line has a slope of less than 1. By sampling in the x-direction, we shall compute y-coordinate that best matches with the ideal line. In Fig. 2.2.5, (15, 20) is the known pixel value, next pixel can be drawn on (16, 20) or (16, 21).
- Thus, for the line with slope $|m| < 1$, if the current pixel is (x_k, y_k) , then next pixel on line would be at $(x_k + 1, y_k)$ or $(x_k + 1, y_k + 1)$.
- For each $(x_k + 1)$ position, algorithm determines on which scan line pixel should be plotted, i.e. y_k or $y_k + 1$?
- The computed y-coordinate for $(x_k + 1)$ position is let's say y . Distance between y and y_k is d_1 and distance between $(y_k + 1)$ and y is d_2 (Refer Fig. 2.2.6).
- From Fig. 2.2.6, it is clear that $d_2 < d_1$ implies line is close to scan line $(y_k + 1)$, so we shall plot pixel at $(x_k + 1, y_k + 1)$ location.
- And if $d_1 < d_2$, the line is close to scan line y_k and hence next pixel remains on the same scan line, and we shall plot pixel at $(x_k + 1, y_k)$.
- Thus, the choice of the next pixel depends on the sign of $(d_1 - d_2)$. If it is positive, plot $(x_k + 1, y_k)$ i.e. pixel in East (E) direction, else plot $(x_k + 1, y_k + 1)$ i.e. Pixel in North-East (NE) direction. This computation needs to be performed recursively till the next endpoint of the line.

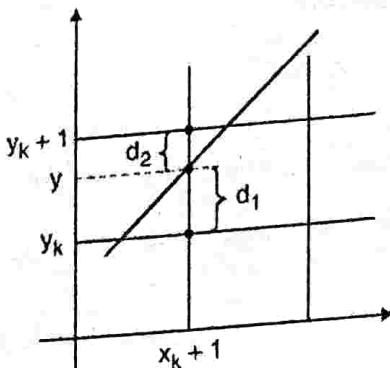


Fig. 2.2.6

- Implicit representation of line is $y = mx + c$. At sampling position $x_k + 1$, corresponding y is computed as,

$$y = m(x_k + 1) + c = m \cdot x_k + m + c$$

$$d_1 = y - y_k = (m \cdot x_k + m + c) - y_k$$

$$d_2 = y_{k+1} - y = y_k + 1 - (m \cdot x_k + m + c)$$

(\because for next scan line, $y_{k+1} = y_k + 1$)

$$\begin{aligned} d_1 - d_2 &= (m \cdot x_k + m + c - y_k) - (y_k + 1 - m \cdot x_k - m - c) \\ &= m \cdot x_k + m + c - y_k - y_k - 1 + m \cdot x_k + m + c \\ &= 2m \cdot x_k + 2m - 2y_k + 2c - 1 \end{aligned}$$

- To simplify the computation, multiply it by Δx ,

$$\begin{aligned} \text{Decision parameter at step } k, p_k &= \Delta x(d_1 - d_2) = \Delta x(2m \cdot x_k + 2m - 2y_k + 2c - 1) \\ &= 2\Delta x \cdot m \cdot x_k + 2\Delta x \cdot m - 2\Delta x \cdot y_k + 2\Delta x \cdot c - \Delta x \end{aligned}$$

Put, $m = \Delta y / \Delta x$ in above equation and simplify,

$$\begin{aligned} &= 2\Delta y \cdot x_k + 2\Delta y - 2\Delta x \cdot y_k + 2\Delta x \cdot c - \Delta x \quad \dots(2.2.1) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + B \end{aligned}$$

Where, $B = 2\Delta y + \Delta x(2c - 1)$. Value of 'B' does not depend on pixel position, so in subsequent computation of p_k , 'B' would be eliminated.

- Decision parameter at step $k + 1$ would be,

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + B$$

- Subtracting p_{k+1} and p_k , we get incremental parameter value,

$$p_{k+1} - p_k = (2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + B) - (2\Delta y \cdot x_k - 2\Delta x \cdot y_k + B)$$

Increment is x direction is always 1, so $x_{k+1} = x_k + 1$.

$$\begin{aligned} p_{k+1} - p_k &= 2\Delta y \cdot (x_k + 1) - 2\Delta x \cdot y_{k+1} - 2\Delta y \cdot x_k + 2\Delta x \cdot y_k \\ &= 2\Delta y \cdot x_k + 2\Delta y - 2\Delta x \cdot y_{k+1} - 2\Delta y \cdot x_k + 2\Delta x \cdot y_k \end{aligned}$$

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x (y_{k+1} - y_k)$$

if $p_k \geq 0$, then select NE pixel and hence set $p_{k+1} = p_k + 2\Delta y - 2\Delta x$, otherwise select E pixel and set $p_{k+1} = p_k + 2\Delta y$

Initial Decision Parameter

The line starts from the end point (x_0, y_0) . Implicit equation of line is, $y = mx + c$. The initial point is always on line. By putting first end point (x_0, y_0) in the implicit representation of the line, we get

$$c = y_0 - mx_0 = y_0 - (\Delta y / \Delta x) \cdot x_0$$

Put this value in Equation (2.2.1).

$$\begin{aligned} \text{Initial decision parameter, } p_0 &= 2\Delta y \cdot x_0 + 2\Delta y - 2\Delta x \cdot y_0 + 2\Delta x \cdot (y_0 - (\Delta y / \Delta x) \cdot x_0) - \Delta x \\ &= 2\Delta y \cdot x_0 + 2\Delta y - 2\Delta x \cdot y_0 + 2\Delta x \cdot y_0 - 2\Delta y \cdot x_0 - \Delta x \\ p_0 &= 2\Delta y - \Delta x \end{aligned}$$

Using these decision parameter values, we can compute the pixels along the line as follow.

If decision parameter at any point is positive, then select NE pixel otherwise select E pixel as the next pixel on line.

2.2.2.2 Algorithm

Steps for Bresenham's line drawing algorithm to draw a line in the first quadrant are described here. Following steps should be performed to draw a line (with $|m| < 1$) using Bresenham's algorithm.

Step 1 : Read two endpoints (x_0, y_0) and (x_1, y_1)

Step 2 : Plot the first pixel (x_0, y_0)

Step 3 : Compute the constants.

$$\Delta x = x_1 - x_0, \Delta y = y_1 - y_0,$$

$$\text{initial decision parameter } p = 2\Delta y - \Delta x$$

Step 4 : if $p \geq 0$ then

$$x = x + 1$$

$$y = y + 1$$

$$\text{Set } p = p + 2(\Delta y - \Delta x) \quad // \text{Select pixel in North-East (NE) direction}$$

else

$$x = x + 1$$

$$\text{Set } p = p + 2\Delta y \quad // \text{Select pixel in East (E) direction}$$

Step 5 : Repeat step 4 Δx times

Note : For line with $|m| < 1$, interchange the role of x and y , i.e. increment y in each iteration and compute corresponding x .

Algorithm for Bresenham's Line Drawing Method

Algorithm for Bresenham's line drawing approach (for $|m| < 1$) is described below.

Algorithm BRESSENHAM_LINE(x_0, y_0, x_1, y_1)

```

 $\Delta x \leftarrow x_1 - x_0$ 
 $\Delta y \leftarrow y_1 - y_0$ 
 $P_0 \leftarrow 2\Delta y - \Delta x$ 
for  $k \leftarrow 0$  to  $\Delta x$  do
    if  $p_k \geq 0$  then
        putPixel( $x_{k+1}, y_{k+1}$ )           // Select pixel in North-East direction
         $p_{k+1} \leftarrow p_k + 2(\Delta y - \Delta x)$ 
    else
        putPixel( $x_{k+1}, y_k$ )             // Select pixel in East direction
         $p_{k+1} \leftarrow p_k + 2\Delta y$ 
    end
end

```

Generalization

Above discussed approach is valid for the line $0 < m < 1$ and $x_1 < x_2$. To generalize this approach, perform following steps.

- If $x_1 > x_2$, swap them
- For lines, $m > 1$, interchange x with y.

2.2.2.3 Pros and Cons

Advantages and disadvantages of Bresenham's line drawing algorithm are stated here:

Advantages

1. It involves only integer calculation.
2. It is faster than DDA.
3. In decision parameter, multiplication by 2 can be implemented in hardware using shift register.
4. Involves cheaper operations like addition and subtraction.
5. More accurate.
6. Bresenham's algorithm does not perform rounding operation.

Disadvantages

1. Bresenham's line drawing approach does not consider the anti-aliasing.
2. It may not produce a smooth line.

2.2.2.4 Examples

Ex. 2.2.8 : Calculate the pixel coordinates of line PQ using Bresenham's algorithm. Where, P = (20, 20) and Q = (10, 12).

Soln.:

Let us assume, $(x_1, y_1) = (20, 20)$ and $(x_2, y_2) = (10, 12)$

$$\Delta x = dx = |20 - 10| = 10,$$

$$\Delta y = dy = |20 - 12| = 8$$

$$\text{Initial decision parameter, } p_0 = 2dy - dx = 16 - 10 = 6$$

$$\Delta E = 2dy = 16 \quad (\text{Increment if E pixel is selected})$$

$$\Delta NE = 2(dy - dx) = -4 \quad (\text{Increment if NE pixel is selected})$$

Table P. 2.2.8

k	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	P_k
0	-	-	10	12	$p_k = p_0 = 2dy - dx = 6$
1	(10, 12)	$p_k \geq 0$, so select NE	$10 + 1 = 11$	$12 + 1 = 13$	$p_k = p_k + \Delta NE = 6 - 4 = 2$
2	(11, 13)	$p_k \geq 0$, so select NE	$11 + 1 = 12$	$13 + 1 = 14$	$p_k = p_k + \Delta NE = 2 - 4 = -2$
3	(12, 14)	$p_k < 0$, so select E	$12 + 1 = 13$	$14 + 0 = 14$	$p_k = p_k + \Delta E = -2 + 16 = 14$
4	(13, 14)	$p_k \geq 0$, so select NE	$13 + 1 = 14$	$14 + 1 = 15$	$p_k = p_k + \Delta NE = 14 - 4 = 10$
5	(14, 15)	$p_k \geq 0$, so select NE	$14 + 1 = 15$	$15 + 1 = 16$	$p_k = p_k + \Delta NE = 10 - 4 = 6$
6	(15, 16)	$p_k \geq 0$, so select NE	$15 + 1 = 16$	$16 + 1 = 17$	$p_k = p_k + \Delta NE = 6 - 4 = 2$
7	(16, 17)	$p_k \geq 0$, so select NE	$16 + 1 = 17$	$17 + 1 = 18$	$p_k = p_k + \Delta NE = 2 - 4 = -2$
8	(17, 18)	$p_k < 0$, so select E	$17 + 1 = 18$	$18 + 0 = 18$	$p_k = p_k + \Delta E = -2 + 16 = 14$
9	(18, 18)	$p_k \geq 0$, so select NE	$18 + 1 = 19$	$19 + 0 = 19$	$p_k = p_k + \Delta NE = 14 - 4 = 10$
10	(19, 19)	$p_k \geq 0$, so select NE	$19 + 1 = 20$	$19 + 1 = 20$	$p_k = p_k + \Delta NE = 10 - 4 = 6$
11	(20, 20)	-	-	-	-

In Table P. 2.2.8, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.8.

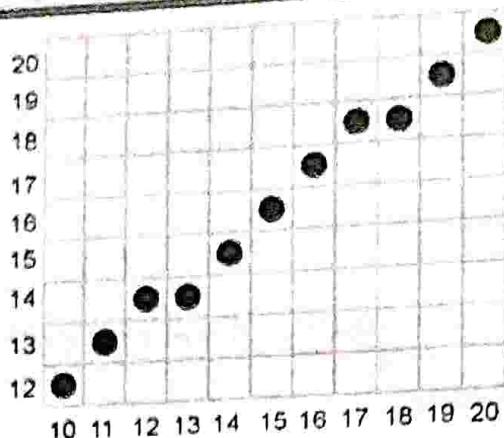


Fig. P. 2.2.8 : Line between (20, 20) to (10, 12) using Bresenham's line drawing approach

Ex. 2.2.9 : Consider line from (4, 4) to (12, 9). Use Bresenham's algorithm to rasterize this line.

Soln. :

Let us assume $(x_1, y_1) = (4, 4)$ and $(x_2, y_2) = (12, 9)$

$$\Delta x = dx = |x_2 - x_1| = |12 - 4| = 8, \Delta y = dy = |y_2 - y_1| = |9 - 4| = 5$$

$$\text{Initial decision parameter, } p_0 = 2dy - dx = 10 - 8 = 2$$

$$\Delta E = 2dy = 10$$

(Increment if E pixel is selected)

$$\Delta NE = 2(dy - dx) = 2(5 - 8) = -6$$

(Increment if NE pixel is selected)

Table P. 2.2.9

k	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	p_k
0	-	-	4	4	$p_k = p_0 = 2dy - dx = 2$
1	(4, 4)	$p_k \geq 0$, so select NE	$4 + 1 = 5$	$4 + 1 = 5$	$p_k = p_k + \Delta NE = 2 - 6 = -4$
2	(5, 5)	$p_k < 0$, so select E	$5 + 1 = 6$	$5 + 0 = 5$	$p_k = p_k + \Delta E = -4 + 10 = 6$
3	(6, 5)	$p_k \geq 0$, so select NE	$6 + 1 = 7$	$5 + 1 = 6$	$p_k = p_k + \Delta NE = 6 - 6 = 0$
4	(7, 6)	$p_k \geq 0$, so select NE	$7 + 1 = 8$	$6 + 1 = 7$	$p_k = p_k + \Delta NE = 0 - 6 = -6$
5	(8, 7)	$p_k < 0$, so select E	$8 + 1 = 9$	$7 + 0 = 7$	$p_k = p_k + \Delta E = -6 + 10 = 4$
6	(9, 7)	$p_k \geq 0$, so select NE	$9 + 1 = 10$	$7 + 1 = 8$	$p_k = p_k + \Delta NE = 4 - 6 = -2$
7	(10, 8)	$p_k < 0$, so select E	$10 + 1 = 11$	$8 + 0 = 8$	$p_k = p_k + \Delta E = -2 + 10 = 8$
8	(11, 8)	$p_k \geq 0$, so select NE	$11 + 1 = 12$	$8 + 1 = 9$	$p_k = p_k + \Delta NE = 8 - 6 = 2$
9	(12, 9)	-	-	-	-

In Table P. 2.2.9, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.9.

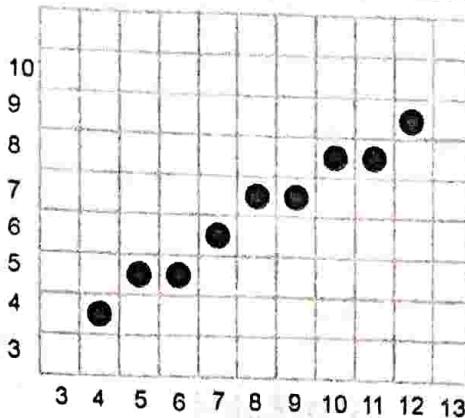


Fig. P. 2.2.9 : Line between (4, 4) to (12, 9) using Bresenham's line drawing approach

Ex. 2.2.10 : Indicate which raster locations would be chosen by Bresenham's algorithm when scan-converting a line from pixel coordinate (1, 1) to pixel coordinate (8, 5).

Soln. :

Let us assume, $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (8, 5)$

$$dx = |8 - 1| = 7, dy = |5 - 1| = 4$$

$$m = dy / dx = 4 / 7 < 1,$$

$$\text{Initial decision parameter, } d = 2dy - dx = 8 - 7 = 1$$

If $d = 0$, we can select any pixel, E or NE. We will select NE here.

$$\Delta E = 2dy = 8$$

$$\Delta NE = 2(dy - dx) = -6$$

Table P. 2.2.10

k	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	d_{new}
0	-	-	1	1	$d = 2dy - dx = 1$
1	(1, 1)	$d > 0$, so select NE	$1 + 1 = 2$	$1 + 1 = 2$	$d + \Delta NE = 1 - 6 = -5$
2	(2, 2)	$d < 0$, so select E	$2 + 1 = 3$	$2 + 0 = 2$	$d + \Delta E = -5 + 8 = 3$
3	(3, 2)	$d > 0$, so select NE	$3 + 1 = 4$	$2 + 1 = 3$	$d + \Delta NE = 3 - 6 = -3$
4	(4, 3)	$d < 0$, so select E	$4 + 1 = 5$	$3 + 0 = 3$	$d + \Delta E = -3 + 8 = 5$
5	(5, 3)	$d > 0$, so select NE	$5 + 1 = 6$	$3 + 1 = 4$	$d + \Delta NE = 5 - 6 = 1$
6	(6, 4)	$d > 0$, so select NE	$6 + 1 = 7$	$5 + 1 = 5$	$d + \Delta NE = 1 - 6 = -5$
7	(7, 5)	$d < 0$, so select E	$7 + 1 = 8$	$5 + 0 = 5$	$d + \Delta E = -5 + 8 = 3$
8	(8, 5)	-	-	-	-

In Table P. 2.2.10, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.10.

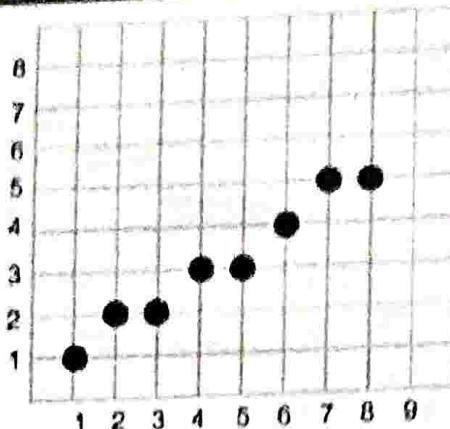


Fig. P. 2.2.10

Ex. 2.2.11 : What are the steps involved in Bresenham's line drawing algorithm for the line $(0, 0)$ to $(-8, -5)$?

Soln.:

To draw a line from left to right, let us assume,

$$(x_1, y_1) = (-8, -5) \text{ and } (x_2, y_2) = (0, 0)$$

$$dx = 0 - (-8) = 8, dy = 0 - (-5) = 5$$

$$\text{Initial decision parameter, } d = 2dy - dx = 10 - 16 = -6$$

As $d < 0$, we select E pixel.

$$\Delta E = 2dy = 10$$

$$\Delta NE = 2(dy - dx) = -6$$

Table P. 2.2.11

K	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	d_{new}
0	-	-	-8	-5	$d = 2dy - dx = -6$
1	(-8, -5)	$d < 0$, so select E	$-8 + 1 = -7$	$-5 + 0 = -5$	$d + \Delta E = -6 + 10 = 4$
2	(-7, -5)	$d > 0$, so select NE	$-7 + 1 = -6$	$-5 + 1 = -4$	$d + \Delta NE = 4 - 6 = -2$
3	(-6, -4)	$d < 0$, so select E	$-6 + 1 = -5$	$-4 + 0 = -4$	$d + \Delta E = -2 + 10 = 8$
4	(-5, -4)	$d > 0$, so select NE	$-5 + 1 = -4$	$-4 + 1 = -3$	$d + \Delta NE = 8 - 6 = 2$
5	(-4, -3)	$d > 0$, so select NE	$-4 + 1 = -3$	$-3 + 1 = -2$	$d + \Delta NE = 2 - 6 = -4$
6	(-3, -2)	$d < 0$, so select E	$-3 + 1 = -2$	$-2 + 0 = -2$	$d + \Delta E = -4 + 10 = 6$
7	(-2, -2)	$d > 0$, so select NE	$-2 + 1 = -1$	$-2 + 1 = -1$	$d + \Delta NE = 6 - 6 = 0$
8	(-1, -1)	$d \geq 0$, so select NE	$-1 + 1 = 0$	$-1 + 1 = 0$	-
9	(0, 0)	-	-	-	-

In Table P. 2.2.11, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.11.

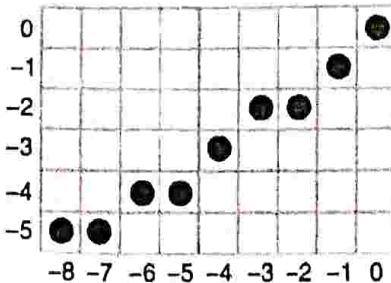


Fig. P. 2.2.11 : Line between $(-8, -5)$ to $(0, 0)$ using the Midpoint line-drawing approach

Ex. 2.2.12: Explain Bresenham line drawing algorithm with proper mathematical analysis and identify the pixel positions along a line between A(10,10) and B(18,16) using it.

MU - May 18, 10 Marks

Soln. :

Let us assume $(x_1, y_1) = (10, 10)$ and $(x_2, y_2) = (18, 16)$

$$dx = |x_2 - x_1| = |18 - 10| = 8,$$

$$dy = |y_2 - y_1| = |16 - 10| = 6$$

Initial decision parameter,

$$P_0 = 2dy - dx = 12 - 8 = 4$$

$$\Delta E = 2dy = 12$$

(Increment if E pixel is selected)

$$\Delta NE = 2(dy - dx) = 2(6 - 8) = -4$$

(Increment if NE pixel is selected)

Table P. 2.2.12

k	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	P_k
0	-	-	10	10	$P_k = P_0 = 2dy - dx = 4$
1	(10, 10)	$P_k \geq 0$, so select NE	$10 + 1 = 11$	$10 + 1 = 11$	$P_k = P_k + \Delta NE = 4 - 4 = 0$
2	(11, 11)	$P_k \geq 0$, so select NE	$11 + 1 = 12$	$11 + 1 = 12$	$P_k = P_k + \Delta NE = 0 - 4 = -4$
3	(12, 12)	$P_k < 0$, so select E	$12 + 1 = 13$	$12 + 0 = 12$	$P_k = P_k + \Delta E = -6 + 12 = 6$
4	(13, 12)	$P_k \geq 0$, so select NE	$13 + 1 = 14$	$12 + 1 = 13$	$P_k = P_k + \Delta NE = 6 - 4 = 2$
5	(14, 13)	$P_k \geq 0$, so select NE	$14 + 1 = 15$	$13 + 1 = 14$	$P_k = P_k + \Delta NE = 2 - 4 = -2$
6	(15, 14)	$P_k < 0$, so select E	$15 + 1 = 16$	$14 + 0 = 14$	$P_k = P_k + \Delta E = -2 + 12 = 10$
7	(16, 14)	$P_k \geq 0$, so select NE	$16 + 1 = 17$	$14 + 1 = 15$	$P_k = P_k + \Delta NE = 10 - 4 = 6$
8	(17, 15)	$P_k \geq 0$, so select NE	$17 + 1 = 18$	$15 + 1 = 16$	-
9	(18, 16)	-	-	-	-

Second column (x_k, y_k) in the above table represents the points on the line.

Ex. 2.2.13 : Write and explain Bresenham's line algorithm and find out which pixel would be turned on for the line with end points (3, 2) to (7, 4) using the same.

Soln. :

Let us assume, $(x_1, y_1) = (3, 2)$ and $(x_2, y_2) = (7, 4)$

$$dx = |7 - 3| = 4, dy = |4 - 2| = 2$$

$$m = dy/dx = 2/4 = 0.5, \text{slope is less than } 1$$

$$\text{Initial decision parameter, } d = 2dy - dx = 4 - 4 = 0$$

If $d = 0$, we can select any pixel, E or NE. We will select NE here.

$$\Delta E = 2dy = 4$$

$$\Delta NE = 2(dy - dx) = -4$$

Table P. 2.2.13

k	(x_k, y_k)	Decision	x_{k+1}	y_{k+1}	d_{new}
0	-	-	3	2	$d = 2dy - dx = 0$
1	(3, 2)	$d = 0$, so select NE	$3 + 1 = 4$	$2 + 1 = 3$	$d + \Delta NE = 0 - 4 = -4$
2	(4, 3)	$d < 0$, so select E	$4 + 1 = 5$	$3 + 0 = 3$	$d + \Delta E = -4 + 4 = 0$
3	(5, 3)	$d = 0$, so select NE	$5 + 1 = 6$	$3 + 1 = 4$	$d + \Delta NE = 0 - 4 = -4$
4	(6, 4)	$d < 0$, so select E	$6 + 1 = 7$	$4 + 0 = 4$	$d + \Delta E = -4 + 4 = 0$
5	(7, 4)	-	-	-	-

In Table P. 2.2.13, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.2.13.

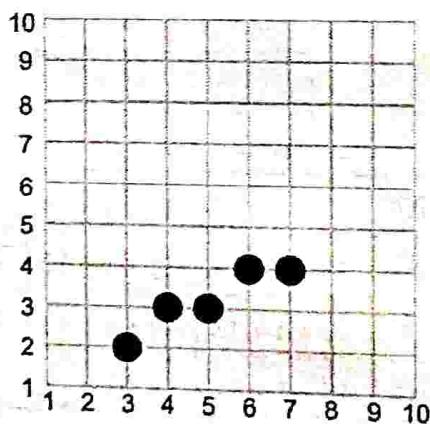


Fig. P. 2.2.13

Ex. 2.2.14 : Write a function Bresenham's line($x_1, y_1, x_2, y_2, \text{type}$) so it draws a line with the specified type. If $\text{type} = 0$, solid line, $\text{type} = 1$ dashed line and $\text{type} = 2$ dotted line.

Soln. :

The solid line plots pixels continuously. Dotted line plots alternate pixel. And we will consider the length of the dash in dashed line 5 pixels. So first we plot 5 pixels followed by a dot and this process is repeated till the end of the line.

Algorithm BRESENHAM LINE (x_1, y_1, x_2, y_2 , type)

```

 $\Delta x \leftarrow x_2 - x_1$ 
 $\Delta y \leftarrow y_2 - y_1$ 
 $d \leftarrow 2\Delta y - \Delta x$                                 // Initial decision parameter
 $\Delta E \leftarrow 2\Delta y$ 
 $\Delta NE \leftarrow 2(\Delta y - \Delta x)$ 
 $x \leftarrow x_1$ 
 $y \leftarrow y_1$ 
Count  $\leftarrow 0$ 
flag  $\leftarrow 1$ 
putPixel (x, y)

while (x < x2) do
    if d < 0 then
        d  $\leftarrow d + \Delta E$ 
    else
        d  $\leftarrow d + \Delta NE$ 
        y  $\leftarrow y + 1$ 
    end
    x  $\leftarrow x + 1$ 

    if type == 0 then                                // Solid Line
        putPixel (x, y)
    end

    if type == 1 then                                // Dashed Line
        if (Count % 5) != 0 then
            putPixel (x, y)
            Count  $\leftarrow Count + 1$ 
        end
    end

    if type == 2 then                                // Dotted Line
        if x%2 == 0 then
            // Skip every odd pixels
        end
    end

```



```

    putPixel(x, y)
end
end
end

```

Ex. 2.2.15 : Write Bresenham's line drawing algorithm to draw dotted line.

Soln. :

Algorithm BRESENHAM LINE(x_0, y_0, x_1, y_1)

$\Delta x \leftarrow x_1 - x_0$

$\Delta y \leftarrow y_1 - y_0$

$P_0 \leftarrow 2\Delta y - \Delta x$

for $k \leftarrow 0$ to Δx do

$x_{k+1} \leftarrow x_k + 1$

 if $p_k < 0$ then

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

 else

$y_{k+1} \leftarrow y_k + 1$

$p_{k+1} \leftarrow p_k + (2\Delta y - 2\Delta x)$

 end

 if $k \% 2 == 0$ then

 putpixel(x_k, y_k)

 end

end

2.2.3 DDA vs. Bresenham Line Drawing Algorithm

Sr. No.	DDA Algorithm	Bresenham's Algorithm
1.	Involves floating-point calculation.	Purely based on integer calculation.
2.	Involves costly operations like multiplication and division.	Involves cheaper operations like addition and subtraction.
3.	Due to floating point operation, it is slower.	It is faster as it involves only integer calculation.
4.	Less accurate.	More accurate.
5.	DDA performs rounding off operation of each pixel.	Bresenham's algorithm does not perform a rounding operation.
6.	Expensive due to extensive multiplication and division operations.	Less expensive as it computes the points on line using addition and subtraction.

2.3 Circle Drawing Algorithm

- The circle is another very useful and important output primitive. Almost all graphics packages provide the functionality to plot and transform the circle.
- Implicit equation of a circle centered at the origin is given by $x^2 + y^2 = r^2$, where r is the radius and (x, y) is any pair of the coordinate on circle boundary which satisfies the above equation.
- If the circle is centered at the point (x_c, y_c) , it should be translated to origin to plot the circle. The equation of the circle, in this case, would be $(x - x_c)^2 + (y - y_c)^2 = r^2$.

2.3.1 Midpoint Circle Drawing Algorithm

2.3.1.1 Working Mechanism

- Principle of midpoint circle drawing approach is identical to the midpoint line-drawing approach.
- Moving down from current pixel (x_k, y_k) , possible point to be plotted is either East pixel E or South-East pixel (SE) as shown in Fig. 2.3.1. Coordinates of midpoint M are $(x_k + 1, y_k - \frac{1}{2})$.

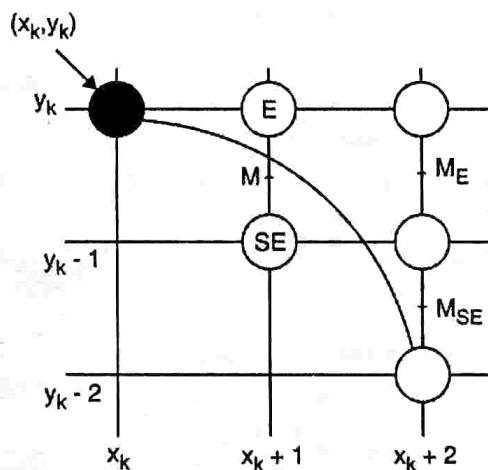


Fig. 2.3.1 : Midpoint circle drawing approach

- We can write circle equation in explicit form $f(x, y) = x^2 + y^2 - r^2$. We will put the value of midpoint coordinates in this function and evaluate its sign. If the function returns 0, means midpoint is on the circle and we can select E or SE as the next pixel. If the function returns positive value, implies midpoint is inside circle and the arc is closer to E pixel so select E as next pixel. And if function returns negative value, implies midpoint is outside circle and the arc is closer to SE pixel so select SE as next pixel.
- Like line drawing algorithm, we will compute decision parameter d to decide the next pixel.

$$d = d_{\text{old}} = f(M) = f\left(x_k + 1, y_k - \frac{1}{2}\right) = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2,$$

If $d = f(M) = 0$, then select E or SE as a next pixel

$d = f(M) > 0$, then select SE as a next pixel

$d = f(M) < 0$, then select E as a next pixel

- As shown in Fig. 2.3.1, if E is selected then, next mid point would be $M_E(x_k + 2, y_k - \frac{1}{2})$ and if SE is selected, next mid point would be $M_{SE}(x_k + 2, y_k - \frac{3}{2})$.

If E is chosen :

$$\begin{aligned}
 d_{\text{new}} &= f(M) = f(M_E) = f\left(x_k + 2, y_k - \frac{1}{2}\right) \\
 &= (x_k + 2)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \\
 &= \left(x_k^2 + 4x_k + 4\right) + \left(y_k - \frac{1}{2}\right)^2 - r^2 \\
 &= \left(x_k^2 + 2x_k + 1\right) + \left(y_k - \frac{1}{2}\right)^2 - r^2 + 2x_k + 3 \\
 &= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 + 2x_k + 3
 \end{aligned}$$

$$\text{But } d_{\text{old}} = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

$$\therefore d_{\text{new}} = d_{\text{old}} + 2x_k + 3$$

We define this increment as ΔE

$$\therefore \Delta E = d_{\text{new}} - d_{\text{old}} = 2x_k + 3$$

This implies, if E is selected, we can compute the value of the next decision parameter just by adding $2x_k + 3$ to the old decision parameter.

If SE is chosen :

$$\begin{aligned}
 d_{\text{new}} &= f(M) = f(M_{SE}) = f\left(x_k + 2, y_k - \frac{3}{2}\right) \\
 &= (x_k + 2)^2 + \left(y_k - \frac{3}{2}\right)^2 - r^2 \\
 &= \left(x_k^2 + 4x_k + 4\right) + \left(y_k^2 - 3y_k + \frac{9}{4}\right) - r^2 \\
 &= \left(x_k^2 + 2x_k + 1\right) + \left(y_k^2 - y_k + \frac{1}{4}\right) - r^2 + (2x_k + 3) + (-2y_k + 2) \\
 &= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 + 2x_k - 2y_k + 5
 \end{aligned}$$

$$\text{But } d_{\text{old}} = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

$$d_{\text{new}} = d_{\text{old}} + 2x_k - 2y_k + 5$$

We define this increment as ΔSE

$$\therefore \Delta SE = d_{\text{new}} - d_{\text{old}} = 2x_k - 2y_k + 5$$

This implies, if SE is selected, we can compute the value of the next decision parameter just by adding $2x_k - 2y_k + 5$ to the old decision parameter.

The initial point of the circle is always known, it is $(0, r)$. The first midpoint would be at $\left(1, r - \frac{1}{2}\right)$ and an initial decision parameter is,

$$\begin{aligned} d_{\text{initial}} &= f\left(1, r - \frac{1}{2}\right) \\ &= (1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2 \\ &= 1 + \left(r^2 - r + \frac{1}{4}\right) - r^2 \\ \therefore d_{\text{initial}} &= \frac{5}{4} - r \end{aligned}$$

Here, d_{initial} introduces the floating-point arithmetic. We are interested in sign, not in magnitude. So we will accept slight modification $d_{\text{initial}} = 1 - r$.

2.3.1.2 Algorithm

Algorithm MIDPOINT_CIRCLE(r)

```

x ← 0
y ← r
d ← 1 - r
EightWaySymmetry (x, y)

while x < y do
    if d < 0 then
        d ← d + 2x + 3
    else
        d ← d + 2x - 2y + 5
        y ← y - 1
    end
    x ← x + 1
    EightWaySymmetry (x, y)
end

```

Routine for eight way symmetry is shown below :

EightWaySymmetry (x, y)

```

putPixel (x, y)
putPixel (x, -y)
putPixel (-x, y)
putPixel (-x, -y)

```

```

putPixel (y, x)
putPixel (y, -x)
putPixel (-y, x)
putPixel (-y, -x)

```

2.3.1.3 Pros and Cons

Advantages

- It is powerful and efficient
- No special programming skill is needed
- This algorithm is used to generate complex graphics on the raster system
- It involves integer calculations only

Disadvantages

- It is time-consuming
- Circle generated using this algorithm is not so smooth due to uneven distance between pixels

2.3.1.4 Example

Ex. 2.3.1 : With the help of the midpoint circle drawing algorithm, find out the coordinates of points that lie on the circle with radius 10 and (15, 5) as center.

Soln. : First, we will derive the pixels for the circle centred at the origin, and then we will translate them by (15, 5).

$$\text{Circle equation } x^2 + y^2 - r^2 = 0$$

$$\text{So, } x^2 + y^2 - 100 = 0$$

We start with $x = 0$ and $y = r = 10$

Initial decision parameter $d = 1 - r = 1 - 10 = -9$

If $d < 0$, select E and $\Delta E = 2x_k + 3$

If $d > 0$, select SE and $\Delta SE = 2x_k - 2y_k + 5$

Table P. 2.3.1

k	(x_k , y_k)	Decision	$x_k + 1$	$y_k + 1$	d_{new}
0	-	-	0	10	$d = 1 - r = 1 - 10 = -9$
1	(0, 10)	$d < 0$, so select E	$0 + 1 = 1$	$10 + 0 = 10$	$d = d + \Delta E = -9 + 3 = -6$
2	(1, 10)	$d < 0$, so select E	$1 + 1 = 2$	$10 + 0 = 10$	$d = d + \Delta E = -6 + 7 = 1$
3	(2, 10)	$d < 0$, so select E	$2 + 1 = 3$	$10 + 0 = 10$	$d = d + \Delta E = 1 + 7 = 8$
4	(3, 10)	$d > 0$, so select SE	$3 + 1 = 4$	$10 - 1 = 9$	$d = d + \Delta SE = 8 - 9 = -1$

k	(x_k, y_k)	Decision	x_k + 1	y_k + 1	d_{new}
5	(4, 9)	d < 0, so select E	4 + 1 = 5	9 + 0 = 9	d = d + ΔE = -3 + 11 = 8
6	(5, 9)	d > 0, so select SE	5 + 1 = 6	9 - 1 = 8	d = d + ΔSE = 8 - 3 = 5
7	(6, 8)	d > 0, so select SE	6 + 1 = 7	8 - 1 = 7	8 >= y, so stop
8	(7, 7)	-	-	-	-

After translation by vector [15 5], we get the points on circle boundary in the first octant as shown below:

Table P. 2.3.1(a)

Points for a circle centered at origin	Points for circle centered at (15, 5)
(0, 10)	(15, 15)
(1, 10)	(16, 15)
(2, 10)	(17, 15)
(3, 10)	(18, 15)
(4, 9)	(19, 14)
(5, 9)	(20, 14)
(6, 8)	(21, 13)
(7, 7)	(22, 12)

By using 8-way symmetry, we can generate pixels in the remaining seven octants.

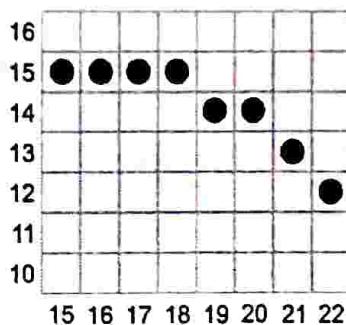


Fig. P.2.3.1 : Circle in the second octant from (15, 15) to (22, 12) using Midpoint circle drawing approach

Ex. 2.3.2 : Apply algorithm and find out points for a circle with radius 8 and centre (0, 0) for one octant only.

Soln. :

We start with $x = 0$ and $y = r = 8$

Initial decision parameter $d = 1 - r = 1 - 8 = -7$

If $d < 0$, select E and $\Delta E = 2x_k + 3$

If $d > 0$, select SE and $\Delta SE = 2x_k - 2y_k + 5$

Table P. 2.3.2

k	(x_k, y_k)	Decision	$x_k + 1$	$y_k + 1$	d_{new}
0	-	-	0	8	$d = 1 - r = 1 - 8 = -7$
1	(0, 8)	$d < 0$, so select E	$0 + 1 = 1$	$8 + 0 = 8$	$d = d + \Delta E = -7 + 3 = -4$
2	(1, 8)	$d < 0$, so select E	$1 + 1 = 2$	$8 + 0 = 8$	$d = d + \Delta E = -4 + 5 = 1$
3	(2, 8)	$d > 0$, so select SE	$2 + 1 = 3$	$8 - 1 = 7$	$d = d + \Delta SE = 1 - 7 = -6$
4	(3, 7)	$d < 0$, so select E	$3 + 1 = 4$	$7 + 0 = 7$	$d = d + \Delta E = -6 + 9 = 3$
5	(4, 7)	$d > 0$, so select SE	$4 + 1 = 5$	$7 - 1 = 6$	$d = d + \Delta SE = 3 - 1 = 2$
6	(5, 6)	$d > 0$, so select SE	$5 + 1 = 6$	$6 - 1 = 5$	-
7	(6, 6)	-	-	-	-

In Table P. 2.3.2, column 2, i.e. (x_k, y_k) are the points to be displayed. They are shown in Fig. P. 2.3.2.

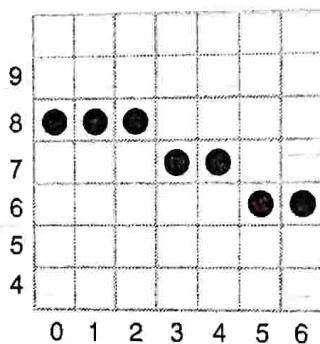


Fig. P. 2.3.2 : Circle in the second octant from (0, 8) to (6, 6) using Midpoint circle drawing approach

Ex. 2.3.3 : Specify the midpoint circle algorithm, using the same, plot the circle whose radius is 8 units and center at (10, 10)

MU - Dec. 18, 10 Marks

Soln. : First, we will derive the pixels for the circle centered at the origin, and then we will translate them to (10, 10).

We start with $x = 0$ and $y = r = 8$

Initial decision parameter $d = 1 - r = 1 - 8 = -7$

If $d < 0$, select E and $\Delta E = 2x_k + 3$

If $d > 0$, select SE and $\Delta SE = 2x_k - 2y_k + 5$

Table P. 2.3.3

k	(x_k, y_k)	Decision	$x_k + 1$	$y_k + 1$	d_{new}
0	-	-	0	8	$d = 1 - r = 1 - 8 = -7$
1	(0, 8)	$d < 0$, so select E	$0 + 1 = 1$	$8 + 0 = 8$	$d = d + \Delta E = -7 + 3 = -4$
2	(1, 8)	$d < 0$, so select E	$1 + 1 = 2$	$8 + 0 = 8$	$d = d + \Delta E = -4 + 5 = 1$

k	(x_k, y_k)	Decision	$x_k + 1$	$y_k + 1$	d_{new}
3	(2, 8)	$d > 0$, so select SE	$2 + 1 = 3$	$8 - 1 = 7$	$d = d + \Delta SE = 1 - 7 = -6$
4	(3, 7)	$d < 0$, so select E	$3 + 1 = 4$	$7 + 0 = 7$	$d = d + \Delta E = -6 + 9 = 3$
5	(4, 7)	$d > 0$, so select SE	$4 + 1 = 5$	$7 - 1 = 6$	$d = d + \Delta SE = 3 - 1 = 2$
6	(5, 6)	$d > 0$, so select SE	$5 + 1 = 6$	$6 - 1 = 5$	-
7	(6, 6)	-	-	-	-

After translation by vector [10 10], we get the points on circle boundary in the first octant as shown below :

Table P. 2.3.3(a)

Points for a circle centered at the origin	Points for a circle centered at (10, 10)
(0, 8)	(10, 18)
(1, 8)	(11, 18)
(2, 8)	(12, 18)
(3, 7)	(13, 17)
(4, 7)	(14, 17)
(5, 6)	(15, 16)
(6, 6)	(16, 16)

By using 8-way symmetry, we can generate pixels in the remaining seven octants.

2.4 Midpoint Ellipse Drawing Algorithm

2.4.1 Working Mechanism

- The slope of tangent at boundary point $(a, 0)$ is zero and tangent at point $(0, b)$ is infinite. As we move along the boundary, slope changes for every pixel. As shown in Fig. 2.4.1, we divide the first quadrant into two parts from the pixel whose slope is -1 .

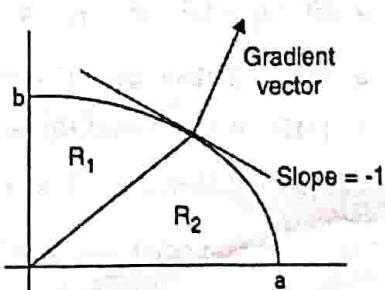


Fig. 2.4.1 : Gradient vector direction for the tangent



- Mathematically ellipse is defined as,

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

Equivalently,

$$b^2 \cdot x^2 + a^2 \cdot y^2 - a^2 \cdot b^2 = 0$$

- The gradient is the perpendicular vector to the slope at given point. It is defined as,

$$\begin{aligned}\text{Grad}(f(x, y)) &= \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j \\ &= \frac{\partial(b^2 \cdot x^2 + a^2 \cdot y^2 - a^2 b^2)}{\partial x} i + \frac{\partial(b^2 \cdot x^2 + a^2 \cdot y^2 - a^2 b^2)}{\partial y} j \\ &= 2b^2 x i + 2a^2 y j\end{aligned}$$

- In region R_1 , the magnitude of j (i.e. Y component) is greater than magnitude of i (i.e. X component). In region 2, converse is true. The point where both have same magnitude, slope would be -1 .
- If we assume the coordinate of point where slope becomes -1 is (x_k, y_k) , then the next midpoint lies at location $(x_k + 1, y_k - \frac{1}{2})$.
- So region switches when the following condition is true.

$$2b^2(x_k + 1)i \geq 2a^2\left(y_k - \frac{1}{2}\right)j$$

- Similarly line and circle drawing algorithm, choice of next pixel is based on the position of mid point is decided by the sign of function. We need to compute decision parameters in both regions.

Region - I : ($dx > dy$)

- As shown in Fig. 2.4.2, from current pixel (x_k, y_k) , we have two choices for the next pixel, East (E) or South East (SE). The midpoint lies at location $(x_k + 1, y_k - \frac{1}{2})$. Let's derive decision parameter by evaluating the ellipse equation $f(x, y) = b^2 \cdot x^2 + a^2 \cdot y^2 - a^2 b^2$ for midpoint.

$$\begin{aligned}d &= d_{\text{old}} = f(M) = f\left(x_k + 1, y_k - \frac{1}{2}\right) \\ &= b^2 \cdot (x_k + 1)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2\end{aligned}$$

If $d = f(M) = 0$, then select E or SE as a next pixel
 $d = f(M) > 0$, then select SE as a next pixel
 $d = f(M) < 0$, then select E as a next pixel

- As shown in Fig. 2.4.2, if E is selected then, next mid point would be $M_E(x_k + 2, y_k - \frac{3}{2})$ and if SE is selected, next mid point would be $M_{SE}(x_k + 2, y_k - \frac{1}{2})$.

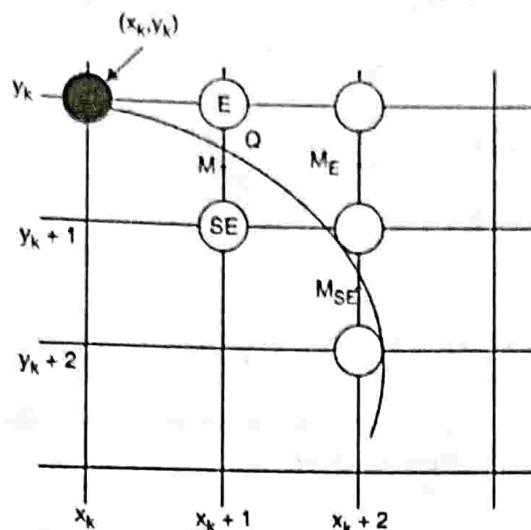


Fig. 2.4.2 : Processing of region 1 in midpoint ellipse drawing approach

If E is chosen

$$\begin{aligned}
 d_{\text{new}} &= f(x_k + 2, y_k - \frac{1}{2}) = b^2 \cdot (x_k + 2)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2 \\
 &= b^2 \cdot \left(x_k^2 + 4x_k + 4\right) + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2 \\
 &= b^2 \left(x_k^2 + 2x_k + 1\right) + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2 + b^2 (2x_k + 3) \\
 &= b^2 (x_k + 1)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2 + b^2 (2x_k + 3)
 \end{aligned}$$

$$\text{But } d_{\text{old}} = b^2 \cdot (x_k + 1)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2$$

$$d_{\text{new}} = d_{\text{old}} + b^2 (2x_k + 3)$$

We define this increment as ΔE

$$\therefore \Delta E = d_{\text{new}} - d_{\text{old}} = b^2 (2x_k + 3)$$

This implies, if E is selected, we can compute the value of the next decision parameter just by adding $b^2 (2x_k + 3)$ to old decision parameter.

If SE is chosen

$$\begin{aligned}
 d_{\text{new}} &= f(x_k + 2, y_k - \frac{3}{2}) \\
 &= b^2 \cdot (x_k + 2)^2 + a^2 \cdot \left(y_k - \frac{3}{2}\right)^2 - a^2 b^2 \\
 &= b^2 \cdot \left(x_k^2 + 4x_k + 4\right) + a^2 \cdot \left(y_k^2 - 3y_k + \frac{9}{4}\right) - a^2 b^2 \\
 &= b^2 \left(x_k^2 + 2x_k + 1\right) + a^2 \cdot \left(y_k^2 - y_k + \frac{1}{4}\right) - a^2 b^2 + b^2 (2x_k + 3) + a^2 \cdot (-2y_k + 2)
 \end{aligned}$$



$$= b^2 \cdot (x_k + 1)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2 + b^2 (2x_k + 3) + a^2 \cdot (-2y_k + 2)$$

$$\text{But } d_{\text{old}} = b^2 \cdot (x_k + 1)^2 + a^2 \cdot \left(y_k - \frac{1}{2}\right)^2 - a^2 b^2$$

$$d_{\text{new}} = d_{\text{old}} + b^2 (2x_k + 3) + a^2 \cdot (-2y_k + 2)$$

We define this increment as ΔSE

$$\Delta SE = d_{\text{new}} - d_{\text{old}} = b^2 (2x_k + 3) + a^2 \cdot (-2y_k + 2)$$

- This implies, if SE is selected, we can compute the value of next decision parameter just by adding $b^2 (2x_k + 3) + a^2 \cdot (-2y_k + 2)$ to old decision parameter.
- If the major and minor axis of ellipse are a and b respectively, the initial point in region 1 would be $(0, b)$, and the next midpoint lies at $\left(1, b - \frac{1}{2}\right)$ initial decision parameter is,

$$d_{\text{initial}} = f\left(1, b - \frac{1}{2}\right) = b^2 (1)^2 + a^2 \left(b - \frac{1}{2}\right)^2 - a^2 b^2 = b^2 + a^2 \left(b^2 - b + \frac{1}{4}\right) - a^2 b^2$$

$$= b^2 + a^2 b^2 + a^2 \left(-b + \frac{1}{4}\right) - a^2 b^2$$

$$\therefore d_{\text{initial}} = b^2 + a^2 \left(-b + \frac{1}{4}\right)$$

- After calculating decision parameter at each pixel, we should also check the value of gradient to see the region change.

Region - II : ($dx < dy$)

As shown in Fig. 2.4.3, in region 2, if the current pixel is (x_k, y_k) , midpoint lies at $\left(x_k + \frac{1}{2}, y_k - 1\right)$, and it will be between South (S) and South East (SE) pixels.

$$\begin{aligned} d &= d_{\text{old}} = f(M) = f\left(x_k + \frac{1}{2}, y_k - 1\right) \\ &= b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 1)^2 - a^2 b^2 \end{aligned}$$

If, $d = f(M) = 0$, then select E or SE as a next pixel

$d = f(M) > 0$, then select S as a next time

$d = f(M) < 0$, then select SE as a next time

As shown in Fig. 2.4.3, if S is selected then, next mid point would be $M_S\left(x_k + \frac{1}{2}, y_k - 2\right)$ and if SE is selected, next mid point would be $M_{SE}\left(x_k + \frac{3}{2}, y_k - 2\right)$.

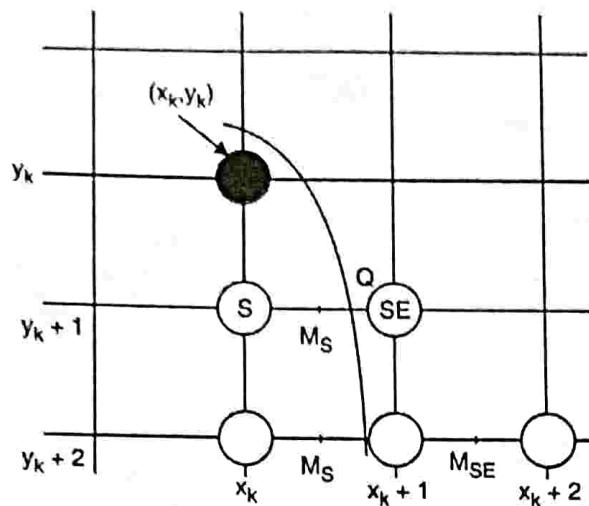


Fig. 2.4.3 : Processing of region 1 in midpoint ellipse drawing approach

If S is chosen

$$\begin{aligned}
 d_{\text{new}} &= f\left(x_k + \frac{1}{2}, y_k - 2\right) = b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 2)^2 - a^2 b^2 \\
 &= b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \left(y_k^2 - 4y_k + 4\right) - a^2 b^2 \\
 &= b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot \left(y_k^2 - 2y_k + 1\right) - a^2 b^2 + a^2 (-2y_k + 3) \\
 &= b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 1)^2 - a^2 b^2 + a^2 (-2y_k + 3)
 \end{aligned}$$

$$\text{But } d_{\text{old}} = b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 1)^2 - a^2 b^2$$

$$d_{\text{new}} = d_{\text{old}} + a^2 (-2y_k + 3)$$

We define this increment as ΔS

$$\therefore \Delta S = d_{\text{new}} - d_{\text{old}} = a^2 (-2y_k + 3)$$

This implies, if E is selected, we can compute the value of next decision parameter just by adding $a^2 (-2y_k + 3)$ to old decision parameter.

If SE is chosen

$$\begin{aligned}
 d_{\text{new}} &= f\left(x_k + \frac{3}{2}, y_k - 2\right) = b^2 \cdot \left(x_k + \frac{3}{2}\right)^2 + a^2 \cdot (y_k - 2)^2 - a^2 b^2 \\
 &= b^2 \cdot \left(x_k^2 + 3x_k + \frac{9}{4}\right) + a^2 \left(y_k^2 - 4y_k + 4\right) - a^2 b^2 \\
 &= b^2 \left(x_k^2 + x_k + \frac{1}{4}\right) + a^2 \cdot \left(y_k^2 - 2y_k + 1\right) - a^2 b^2 + b^2 (2x_k + 2) + a^2 (-2y_k + 3) \\
 &= b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 1)^2 - a^2 b^2 + b^2 (2x_k + 2) + a^2 (-2y_k + 3)
 \end{aligned}$$

$$\text{But } d_{\text{old}} = b^2 \cdot \left(x_k + \frac{1}{2}\right)^2 + a^2 \cdot (y_k - 1)^2 - a^2 b^2$$



$$d_{\text{new}} = d_{\text{old}} + b^2(2x_k + 2) + a^2(-2y_k + 3)$$

We define this increment as ΔSE

$$\therefore \Delta SE = d_{\text{new}} - d_{\text{old}} = +b^2(2x_k + 2) + a^2(-2y_k + 3)$$

This implies, if SE is selected, we can compute the value of next decision parameter just by adding $b^2(2x_k + 2) + a^2(-2y_k + 3)$ to old decision parameter.

2.4.2 Algorithm

Midpoint ellipse drawing algorithm is described here :

Algorithm MIDPOINT_ELLIPSE (a, b)

$x \leftarrow 0, y \leftarrow b$ //Processing in region 1

$d_1 \leftarrow b^2 - a^2b + a^2/4$

FourWaySymmetry (x, y)

while $a^2\left(y - \frac{1}{2}\right) > b^2(x + 1)$ do

if $d_1 < 0$ then

$d_1 \leftarrow d_1 + b^2(2x + 3)$

$x \leftarrow x + 1$

else

$d_1 \leftarrow d_1 + b^2(2x + 3) + a^2(-2y + 2)$

$x \leftarrow x + 1$

$y \leftarrow y - 1$

end

FourWaySymmetry (x, y)

end.

//Processing in region 2

$d_2 \leftarrow b^2\left(x + \frac{1}{2}\right) - a^2(y - 1)^2 - a^2b^2$

while $y > 0$ do

if $d_2 < 0$ then

$d_2 \leftarrow d_2 + b^2(2x + 2) + a^2(-2y + 3)$

$x \leftarrow x + 1$

$y \leftarrow y - 1$

else

$d_2 \leftarrow d_2 + a^2(-2y + 3)$

```

y ← y - 1
end
FourWaySymmetry (x, y)
end

```

FourWaySymmetry routine is shown below :

```
FourWaySymmetry (x, y)
```

```
Putpixel (x, y)
```

```
Putpixel (x, -y)
```

```
Putpixel (-x, y)
```

```
Putpixel (-x, -y)
```

2.4.3 Pros and Cons

Advantages

- It involves only integer calculations
- It is easy to implement

Disadvantages

It is time consuming

2.4.4 Example

Ex. 2.4.1 : Plot the points for midpoint ellipse with $r_x = 3$ and $r_y = 5$ for region 1.

MU - May 19, 10 Marks

Soln. :

In region 1, we starts drawing from the point $(0, r_y) = (0, 5)$.

We keep moving till the condition $2x_{k+1} \cdot r_y^2 \geq 2y_{k+1} \cdot r_x^2$ is false

$$\text{Initial decision parameter : } d_0 = r_y^2 - r_x^2 \cdot r_y + r_x^2 / 4 = 25 - 45 + 9/4 = -18$$

k	(x_k, y_k)	d_k	(x_{k+1}, y_{k+1})	$2x_{k+1} \cdot r_y^2$	$2y_{k+1} \cdot r_x^2$
0	(0, 5)	$d = 25 - 45 + 9/4 = -18$	(1, 5)	50	90
1	(1, 5)	$d = d + r_y^2 (2x + 3) = -18 + 25 (2*0 + 3) = -18 + 75 = 57$	(2, 4)	100	72
2	(2, 4)	$2x_{k+1} \cdot r_y^2 \geq 2y_{k+1} \cdot r_x^2$ is false, so stop the computation			

Points in column (x_k, y_k) in above table will be plotted in region 1.



2.5 Aliasing

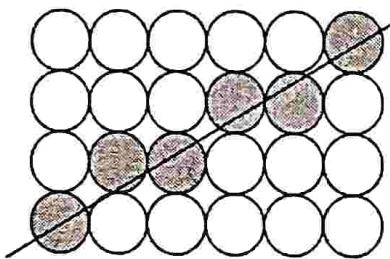
Q. What is aliasing and Explain any one anti-aliasing method.

MU - May 18, Dec. 18, May 19, 5 Marks

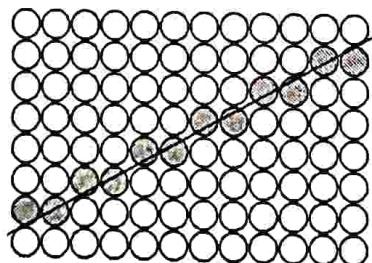
Q. What is aliasing, how it affects the appearance of an object ?

MU - Dec. 19, 5 Marks

- When straight line (except horizontal and vertical) lines are drawn on monitor, it appears zig-zag. This effect is Aliasing happens due to poor algorithm or hardware limitations.
- Zig-zag lines on screen give the illusion of smooth line, that effect is called **anti-aliasing**. Line or curve on paper looks smooth and continuous, but when they are rendered on monitor screen, creates discrete and zig-zag appearance due to grid structure of monitor display.
- We can fool the human eye by applying a certain technique which reduces this aliasing effect. Aliasing effect becomes significant when we enlarge or shrink the size of shape.
- We can achieve anti-aliasing either by increasing the screen resolution or by applying smart algorithms to compute the pixel locations on the shape boundary.
- Fig. 2.5.1(a) and (b) shows the line on low and high resolution respectively. If we increase the resolution twice, numbers of pixels get multiplied by four. Line interpolates more number of pixels with more zig-zags, but the magnitude of these zig-zags would be smaller than the previous one, so line appears smooth. However, this smoothness comes at the cost of memory, time to generate line and CPU processing.



(a) A line on a low-resolution display



(b) A line on a high-resolution display

Fig. 2.5.1 : Effect of resolution on aliasing

- There is an alternative way to reduce the zig-zag pattern. On system which supports multiple bits per pixel, we can adjust the intensity of pixel by setting its intensity value.
- By changing this, we can get rid of the aliasing effect. Anti-aliasing methods can be classified as,
 - o Super sampling /post-filtering
 - o Area sampling/pre filtering

2.6 Anti-aliasing

Q. What is anti-aliasing.

MU - May 19, 5 Marks

Q. Explain any two Antialiasing methods.

MU - Dec. 19, 5 Marks

- We can achieve anti-aliasing either by increasing the screen resolution or by applying smart algorithms to compute the pixel locations on the shape boundary.
- Detail classification of anti-aliasing methods is shown in Fig. 2.6.1.

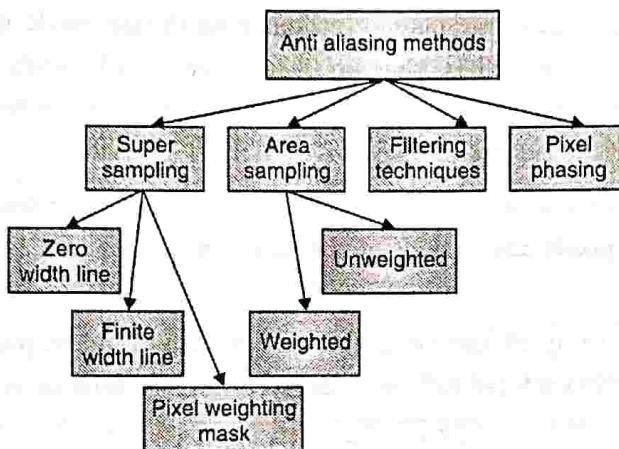


Fig. 2.6.1

2.6.1 Pre Filtering

- It is also known as **area sampling**.
- In this technique, intensity of pixel is determined by the area of pixel covered by the object.
- If object covers the entire pixel then it is illuminated with full intensity, otherwise it is illuminated with intensity proportional to the area covered by object.

1. Unweighted Area Sampling

- As we discussed in mid-point line drawing algorithm, sometimes line passes through two pixels, and algorithm illuminates the pixel which is closer to line.
- Unweighted area sampling is extension of this concept, which reduces the aliasing effect. This method illuminates both the pixels with different intensities.
- The pixel which covers the larger area of line is assigned higher intensity, and pixel which occupies less area of line is assigned lower intensity. Unweighted area sampling assigns the same intensity to two pixels covering same area irrespective of their distance from the center of pixel.

2. Weighted Area Sampling

- In weighted area sampling, pixel intensity depends on its distance from center of pixel. In previous method, small area in the corner of the pixel contributes same as the pixel which covers equal area near the pixel center.



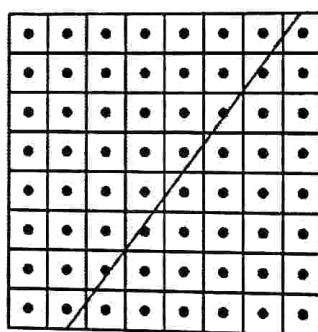
- Remedy to this problem is weighted area sampling. In this technique, intensity of pixel is function of two parameters, one is the area covered by pixel and other is the distance of area from the center of the pixel.
- So the area near to pixel center is assigned higher intensity compare to the farthest area.

2.6.2 Post Filtering

Q. What is aliasing and explain any one anti-aliasing method.

MU - May 18, Dec.18, 5 Marks

- Post filtering is also known as super sampling technique.
- In this method, single-pixel is mapped to imaginary higher resolution mask, each mask location is assigned weight as per the pixel portion covered by that mask. In other words, each pixel itself is divided into an imaginary grid and each grid location is assigned weight according to area covered by the pixel. Average of intensity of that imaginary grid is replaced back to original pixel location.
- In Fig. 2.6.2(a), original line on the screen is shown. It is very much clear that line does not interpolate all pixels perfectly. Some of the pixels are on edge of the line, some are on the line while some are missing the line.
- Fig. 2.6.2(b) shows the super-sampled and enhanced view of some random pixel. We have divided the pixel in 3×3 imaginary grid. Value in each cell indicates the percentage of area covered by line pixel.



(a) Line with the original pixel position

1/3	2/3	1
3/5	1	0
2/3	5/7	0

(b) View of super-sampled pixel

Fig. 2.6.2 : Supersampling

2.6.3 Super Sampling

1. Super Sampling Considering Zero Line Width

- Each pixel is divided in to grid of sub pixels. In Fig. 2.6.3, pixel is divided in to 3×3 sub pixel. Intensity of original pixel is set proportional to number of sub pixels from which line passes. Intensity of pixel is function of count of sub pixels overlapping the line.

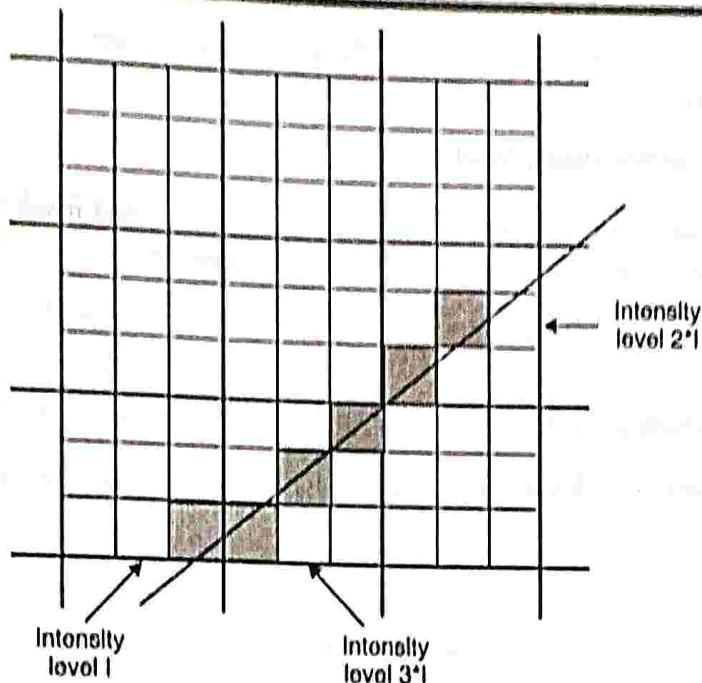


Fig. 2.6.3 : Super sampling considering zero width line

- In Fig. 2.6.3, line intersects only one sub pixel, so intensity of that pixel is set to level I. Similarly, line passes through 3 sub pixels from bottom-middle pixel, so the intensity of that pixel is set to level $3*I$.
- Higher the number of sub pixels, we will have more number of intensity levels. With 4×4 grid, 16 intensity levels are possible, with 5×5 grid, 25 intensity levels are possible. Bigger the size of grid, lesser the effect of aliasing.

2. Super Sampling Considering Finite Width Line

- This method super samples the pixel by setting pixel's intensity proportional to the number of sub pixels inside the polygon. If lower left corner of sub pixel is inside the polygon boundary, then it is considered as inside polygon.

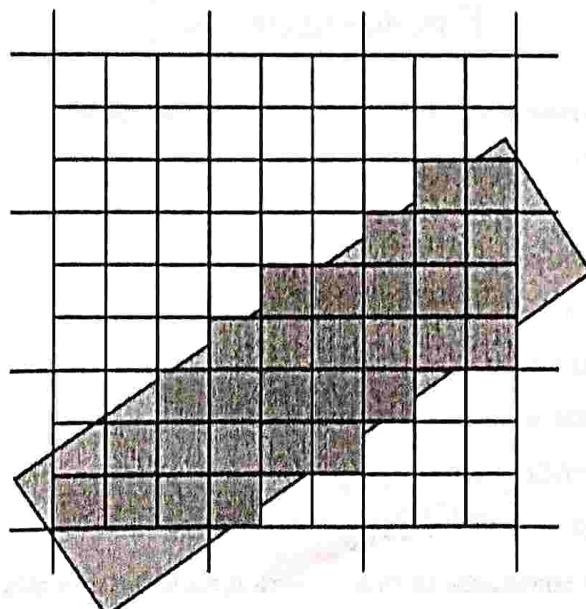


Fig. 2.6.4 : Super sampling considering finite width line



- In this method too, number of intensity levels depends on grid size. We can also interpolate background color in multi-color environment.

3. Super Sampling with Pixel Weighting Mask

- This technique employs the concept of image processing. It uses weighted mask to compute the intensity of pixel. It gives more weight to center pixel. It follows the principle of Gaussian.
- As we move further from the center of mask, weight reduces. We can customize the mask coefficient to change the weight of pixels.
- Mask is put on the pixel which is on object boundary and the weighted sum of the intensity is computed.
- Averaged intensity is replaced with the center pixel. Such mask can be overlapped with neighbour pixel to reduce the stripping effect.

1	2	1
2	4	2
1	2	1

Fig. 2.6.5 : Mask for supersampling

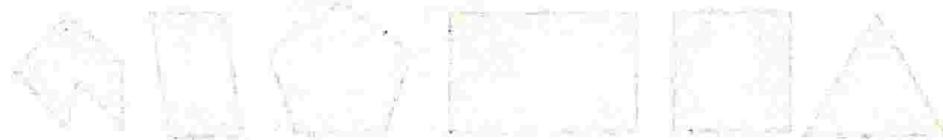
2.6.4 Pixel Phasing

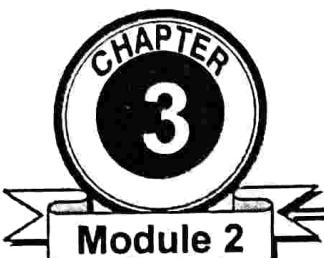
- Another approach to reducing effect of aliasing is to shift the pixel position from its actual position. This technique is applicable when an electron gun can address the sub-pixel region.
- Subpixel position is shifted to form the smooth boundary of object.

Review Questions

- Q. 1 Explain the way of computing pixel positions using the slope-intercept formula.
- Q. 2 Explain the explicit formula of line drawing.
- Q. 3 Define the line segment.
- Q. 4 Discuss the characteristics of the ideal line.
- Q. 5 Explain and write steps for DDA line drawing algorithm.
- Q. 6 Enlist the properties of an ideal line.
- Q. 7 What are the advantages of DDA line drawing algorithm?
- Q. 8 What are the disadvantages of DDA line drawing algorithm?
- Q. 9 Explain with the help of illustration, Bresenham's line drawing algorithm.
- Q. 10 Derive the decision parameter expressions for Bresenham's line drawing algorithm.
- Q. 11 Write the Bresenham's line drawing algorithm and explain how it is better than the DDA algorithm for line generation.

- Q. 12 Mention the steps for Bresenham's line drawing algorithm.
- Q. 13 Enlist advantages and disadvantages of Bresenham's line drawing algorithm.
- Q. 14 State merits and demerits of Bresenham's line drawing algorithm.
- Q. 15 Write an algorithm for Bresenham's Line drawing method.
- Q. 16 Compare DDA and Bresenham's line drawing algorithm.
- Q. 19 Write an algorithm for midpoint circle drawing algorithm.





Filled Area Primitives

Syllabus

Filled Area Primitive : Scan line Polygon Fill algorithm, Inside outside tests, Boundary Fill and Flood fill algorithm.

3.1 Polygon

3.1.1 Introduction to Polygon

Many times basic primitives like point, line or triangles are not sufficient to describe the arbitrary shape. Real-world objects are often more complex. The best way to describe them is to use polygons. In this section, we will study different types of polygons and the methods to fill them.

3.1.2 Types of Polygon

- **Polygon** is a closed figure made up of connected lines. Polygon has many sides and is represented by line segments. Line segments forming the boundary of the polygon are called **edges** of the polygon and the endpoints of edges are known as **vertices** of the polygon.
- Polygon is always closed figure; hence less than three sides cannot form the polygon. Triangle is the polygon with a **minimum number of edges**. The circle is closed figure but it does not have edges, so it is not a polygon. Few examples of polygons are shown in Fig. 3.1.1.



Fig. 3.1.1 : Polygon with different shapes and sides

- We can classify polygon in one of the following categories :
 - o Convex polygon
 - o Concave polygon
 - o Complex polygon

3.1.2.1 Convex Polygon

- In a convex polygon, the angle between any consecutive pair of edges is always less than 180° .
- For a convex polygon, if we select any random two points inside the polygon and if we join them through a line; all the points on the line are certainly inside the polygon.
- Examples of the convex polygon are shown in Fig. 3.1.2.

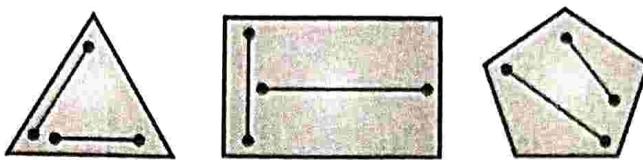


Fig. 3.1.2 : Example of convex polygons

3.1.2.2 Concave Polygon

- In a concave polygon, the angle between at least one pair of consecutive edges is greater than 180° .
- For a concave polygon, there always exists a pair of vertices inside the polygon, when joined through a line segment, part of line segment falls outside the polygon.
- Examples of the concave polygon are shown in Fig. 3.1.3.

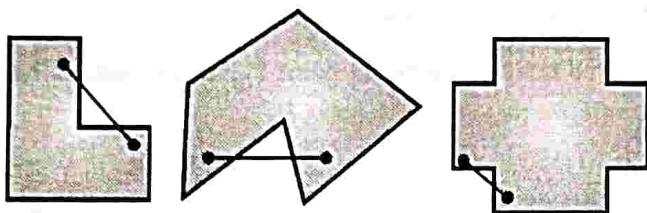


Fig. 3.1.3 : Examples of concave polygons

3.1.2.3 Complex Polygon

- Polygons which are neither convex nor concave are called a complex polygon. Complex polygons are self-intersecting or overlapping.
- Examples of a complex polygon are shown in Fig. 3.1.4

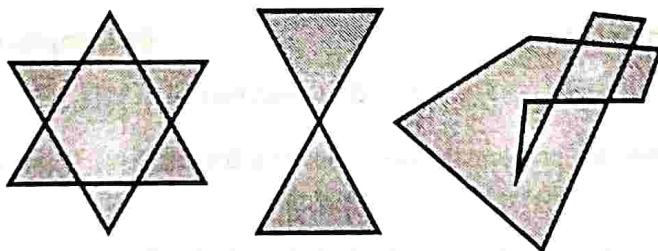


Fig. 3.1.4 : Examples of complex polygons

3.2 Inside Outside Tests

Q. Explain the inside-outside test used in filling algorithm.

MU - Dec. 18, 5 Marks

Q. Write a short note on the inside-outside test.

MU - May 19, 10 Marks

Q. What is the purpose of Inside-Outside Test, explain any one method.

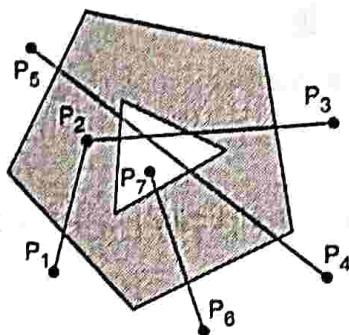
MU - Dec. 19, 5 Marks

During polygon filling, only those pixels should be filled which are inside the polygon. Deciding membership of pixel is very important in polygon filling. Accurate detection of whether the point is inside or outside the polygon is crucial. We will discuss two methods to decide if a pixel is inside the polygon.

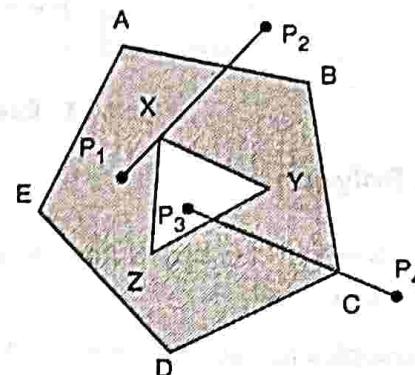
- Even-odd method
- Winding number method

3.2.1 Even-Odd Test

- This method constructs a line from a given point to the known point outside the polygon and checks the intersection of a line with polygon boundary.
- Let's check the visibility of point P_2 (Fig. 3.2.1(a)). Draw a line from P_2 to known exterior point to a polygon in any direction and count number of intersections of line with polygon edges.
- If count is odd, then the point is inside polygon otherwise it is outside of the polygon.
- Line P_2P_1 intersects polygon edges only once, while line P_2P_3 intersects polygon edge thrice, in both the cases count is odd, means the point is inside the polygon.
- Consider the point P_4 , draw a line to known exterior point P_5 , it intersects polygon boundary four times, so P_4 is considered as an exterior point. Same is true for point P_7 .



(a) Simple case



(b) Complex case

Fig. 3.2.1 : Odd-even test

- This rule is very simple, but the problem arises when the line crosses the common vertex shared by two adjacent edges.
- Consider Fig. 3.2.1(b), line P_3P_4 intersects the outer boundary of the polygon at common vertex shared by edges BC and CD. So even though P_3P_4 intersects three polygon edges (YZ, BC and CD), P_3 is an exterior point.
- The solution to this problem is to count intersection as a single intersection if both edges are on the opposite side of the line. For example, CD and BC in case of line P_3P_4 .
- If both edges are on the same side of the line segment, then count it as two intersections.
- For example, XY and XZ are on the same side of line P_1P_2 , so count it as two intersection points.
- It is easy to understand from Fig. 3.2.2, where to count once (odd count) and where to count twice (even count).

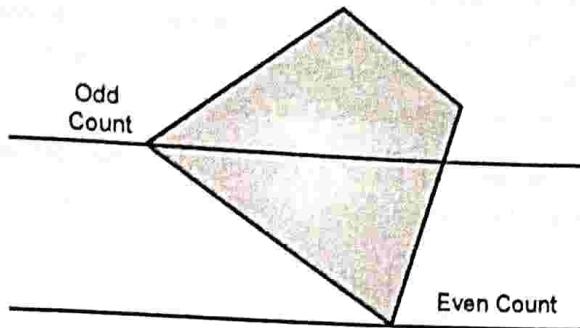


Fig. 3.2.2 : Remedy to shared edges

3.2.2 Winding Number Rule

- Winding number method is another way to define the visibility of pixel.
- Like the even-odd method, draw a straight line from point to be examined to any known exterior point.
- Visit outermost boundary of the polygon in an anti-clockwise direction and inner boundary (if exists) in a clockwise direction.
- Find the intersection points of line with polygon boundaries and check how many times polygon edge crosses the line while traversing from left to right and right to left.

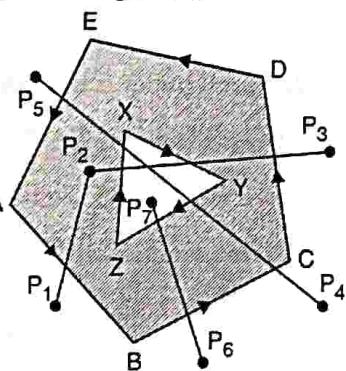


Fig. 3.2.3 : Winding number method

- Take the difference between these two counts, if the difference is zero, the point is outside the polygon, otherwise, it is considered as an inner point.
- Consider the point P_2 , (Refer Fig. 3.2.3) which is joined with exterior point P_1 .
- Polygon edge AB intersects P_1P_2 while traversing from left to right. So point P_2 is the interior point.
- Edge YZ intersect line P_6P_7 from right to left and edge BC intersects it while traversing from left to right. The difference of intersection count is zero and hence P_7 is considered as an outside pixel.
- Similarly edges EA and YZ intersect line P_4P_5 while traversing from right to left and edges ZX and BC intersects P_4P_5 while traversing from left to right. The difference is zero, so point P_4 is outside the polygon.

3.3 Polygon Filling

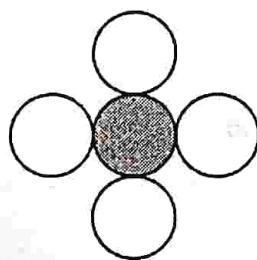
3.3.1 Introduction

- Most of the graphics packages provide area filling primitives. There are two common approaches for area filling in a raster system. One way is to determine the visible span of each scan line of the polygon. This method is used to fill simple shapes like ellipse, circle, rectangle etc. Scan line polygon filling approach is the example of this kind of area filing method.

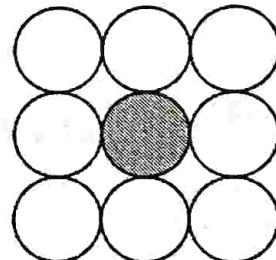
- Another approach is to start with any interior pixel and keep filling region outwards until it hits the boundary. This approach is applicable for filling shapes with complex boundaries and interactive painting system. Flood fill and boundary fill fall in the second category.
- Polygon filling algorithms fill the background of polygons. Most popular algorithms of polygon filling are listed and discussed in the following section.
- Pixel level algorithms
 - o Seed fill algorithms
 - (i) Boundary fill algorithm
 - (ii) Flood fill algorithm
 - o Edge fill algorithm
 - o Fence fill algorithm
- Geometry level algorithms
- Scan line polygon filling algorithm

3.3.2 Pixel Connectivity

- Connectivity defines the relationship between pixel under consideration and its neighbor pixels.
- Most common and useful types of connectivity are shown in Fig. 3.3.1.
- Pixel is called four connected if it has four neighbors and it is called 8-connected if it has eight neighbors.
- Fig. 3.3.1 defines the 4-connected and 8-connected component.



(a) 4 Connected



(b) 8 Connected

Fig. 3.3.1 : Pixel connectivity

3.4 Boundary Fill

3.4.1 Working Mechanism

- Boundary fill algorithm starts with some interior pixel of a polygon, called seed pixel and keep filling neighbor pixels in outward directions until the boundary color is encountered.
- Boundary fill algorithm starts with three parameters: interior point (x, y), fill color and boundary color.
- This approach retrieves the color of the current pixel and compares it with fill color and boundary color.

- If the color of the current pixel is neither fill color nor boundary color, then fill it with the fill color and make a recursive call, otherwise skip the pixel under consideration.
- Neighbour pixels are approached using 4-connectivity or 8-connectivity.
- Sometimes 4-connectivity fails to paint the entire region. 8-connectivity is time consuming and memory intensive.
- This method is useful in interactive painting packages, where the selection of interior pixel can be done very easily using input device like a mouse.
- To create a solid region, set the fill color to boundary color, so that boundary and interior region becomes indistinguishable after filling.
- Recursively this algorithm checks all pixels in given polygon and fills them if not already filled.

3.4.2 Algorithm

- Algorithm for boundary fill method is described here:

Algorithm BOUNDARY_FILL(x, y, fillColor, boundaryColor)

```
// fillColor: Color to be filled in polygon
// boundaryColor: Color of polygon boundary

Current ← getColor (x, y)      // Retrieve the color of current pixel
if   current ≠ fillColor && current ≠ boundaryColor then
    putPixel(x, y, fillColor)
    BOUNDARY_FILL (x + 1, y, fillColor, boundaryColor)
    BOUNDARY_FILL (x - 1, y, fillColor, boundaryColor)
    BOUNDARY_FILL (x, y + 1, fillColor, boundaryColor)
    BOUNDARY_FILL (x, y - 1, fillColor, boundaryColor)
end
```

- This algorithm may not work properly if some of the pixels are already filled, because algorithm returns when the current pixel is either boundary pixel or it is filled.
- To avoid this, we shall set the color of all interior pixels to background color.

3.4.3 Pros and Cons

Advantages

- Simple.
- Easy to implement.
- Works for any type of polygons.

Disadvantages

- Require extensive stacking due to recursion.
- Does not work if the boundary is specified with multiple colors.
- Need to specify seed point contained within the polygon.
- Polygon should be filled with the background color.
- Require more memory.
- Not suitable for the large polygon.

Filling the region with 4 and 8 connectivity :

- Filling of polygon depends on types of connectivity employed. The 4-connected approach does four recursive calls for each of its four nearest neighbors.
- The 8-connected approach makes eight calls. The 4-connected approach requires less memory but sometimes fails to fill certain parts of the polygon.
- In Fig. 3.4.1, if the 4-connected approach is used then it terminates after reaching to pixel marked 'x'. It cannot jump into another part of the polygon. The 8-connected approach can jump diagonally too. Hence it would be able to fill both the regions of the polygon.

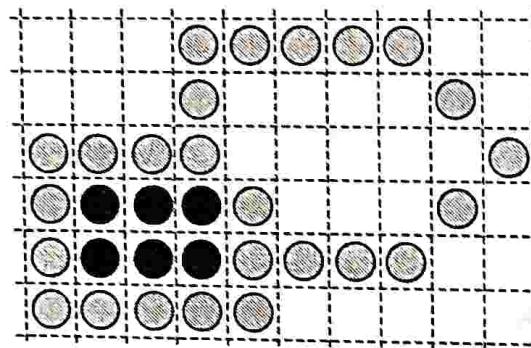


Fig. 3.4.1

3.4.4 Example

Ex. 3.4.1 : Which algorithm cannot be used to fill region R2 which is bounded by Blue color and Red color boundary, justify.

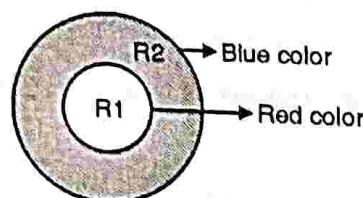


Fig. P. 3.4.1

Soln. :

- Recursive call of boundary fill algorithm returns when a current pixel is either already filled or it has boundary color. It takes fill color and one boundary color as an argument.

- Boundary fill algorithm can either take blue or red color as an argument. Hence, for the region with multiple boundary colors cannot be filled using boundary fill algorithm.
- Whereas, the flood fill algorithm fills pixel if it has a background color. It does not check the boundary color. Hence, it can manage any number of boundary colors.

3.5 Flood Fill

Q. Write the flood fill approach for 8 connected method.

MU - May 19, 5 Marks

3.5.1 Working Mechanism

- Many realistic objects have different colored boundaries. Boundary fill algorithm cannot fill the polygon with multi-colored boundary.
 - Flood fill algorithm can handle this issue.
 - The algorithm starts with the interior pixel, fill color and old color. If the color of the current pixel is the same as old color then it fills the pixel with fill color and examines its neighbors recursively.
 - Like boundary fill algorithm, if the interior pixel is already filled with some other color, flood fill algorithm fails.
- To handle this problem, we shall first paint all interior pixels with background (old) color.

3.5.2 Algorithm

```

Algorithm FLOOD_FILL (x, y, fillColor, oldColor)
current ← getColor (x, y)
if   current ≠ oldColor then
    putPixel (x, y, fillColor)
    FLOOD_FILL (x + 1, y, fillColor, oldColor)
    FLOOD_FILL (x - 1, y, fillColor, oldColor)
    FLOOD_FILL (x, y + 1, fillColor, oldColor)
    FLOOD_FILL (x, y - 1, fillColor, oldColor)
end

```

3.5.3 Pros and Cons

Advantages

- Simple.
- Easy to implement.
- Boundary fill algorithm cannot handle the object with multi-color boundary, whereas flood fill can.

Disadvantages

- Require extensive stacking.



- Slow in nature
- Not suitable for large polygon

3.6 Optimization

- Boundary fill and flood fill, both are recursive approaches. On every call, they make other four or eight calls to the neighbours of the current pixel, this creates extensive stacking. The system runs out of memory if polygon to be filled is very large.
- We will discuss a more efficient method which will minimize the recursive calls. Instead of pushing all neighbors on the stack, just push neighbor scan line numbers on the stack and fill the entire span of the current scan line.
- A sequence of figures shown in Fig. 3.6.1 explains the concept. In Fig. 3.6.1(a), random seed pixel is selected. The span of the current scan line is filled up and addresses of its upper and lower scan line are pushed on to the stack.
- The procedure always pops topmost scan line number from the stack, fills it horizontally and pushes them above and below scan line numbers if they are not already processed. This approach greatly simplifies the processing and reduces memory requirement.

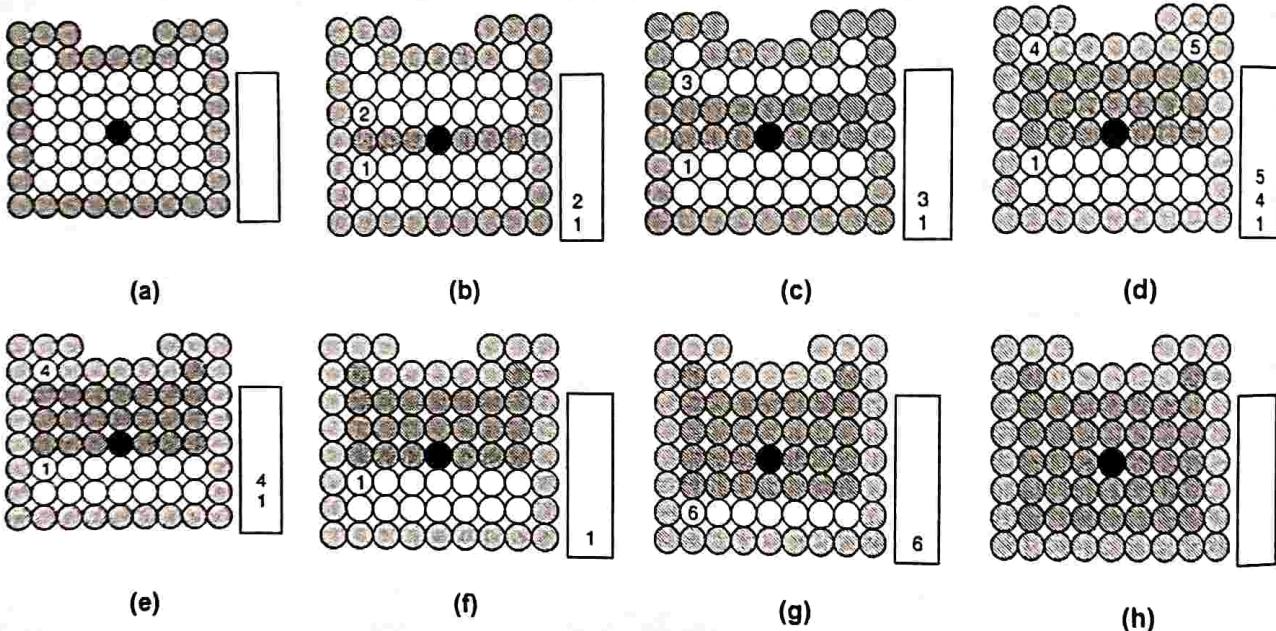


Fig. 3.6.1 : Optimized seed fill approach

3.7 Scan Line Fill

Q. Explain Scan line polygon fill algorithm with the help of suitable diagrams

MU - Dec. 19, 10 Marks

3.7.1 Working Principle

- Seed fill approaches are memory and time intensive due to their recursive structure. Computation can be reduced by determining the entire visible span instead of checking the visibility of individual pixel.

- Scan line fill algorithm computes the visible span of each scan line within polygon boundaries. It works with convex, concave, self-intersecting and polygon with holes.

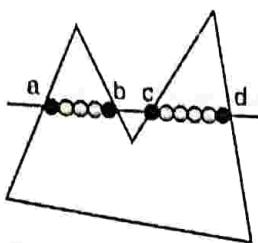


Fig. 3.7.1 : Visible span of the scan line

- The scan line intersects polygon boundary at a, b, c and d (Refer Fig. 3.7.1). We shall set appropriate values of all intersection points to fill the polygon properly.
- Midpoint line clipping algorithm can be used to determine the intersection point of the scan line with the edges.
- This approach selects the pixel which is closer to the edge, irrespective of on which side of polygon it lies. If the edge is shared between two polygons, and the outer pixel is selected for drawing, this would look odd if a neighbor polygon has a different color. So even if the exterior pixel is closer to the edge, we should reject it for better rasterization.

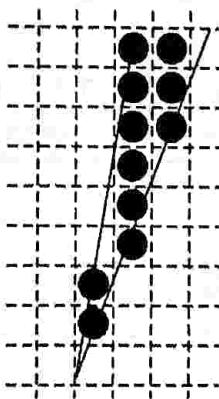


Fig. 3.7.2 : Silver

- A simple solution to this problem is to round up the intersection pixel coordinate if scan line is entering in polygon region, and round down if it is leaving the polygon.
- Scan line polygon filling approach works as follow :
 - o Find intersections of scan line with polygon edges.
 - o Sort all intersections on their x-coordinate.
 - o Fill the span using a pair of sorted intersection points.
- When scan line intersects the polygon boundary, parity is odd and when it intersects again parity becomes even. Fill the span between odd to even parity. Initially, parity would be even.



- For example, in Fig. 3.7.1 parity is initially even. When scanline intersects the polygon edge at point a, parity becomes odd. When the scanline intersects the edge at point b, parity becomes odd. So fill the span from a to b. Similarly, parity will be odd and even for points c and d so span from c to d will be filled up but span b to c will not.
- The problem arises when the scanline intersects the shared vertex. This problem can be handled in a similar way to the odd-even test. If shared edges are on the same side of the scan line, parity is set to even and if shared edges are on the opposite side of scanline then set the parity to odd.
- Horizontal edge is the special case. Parity changes on every pixel for horizontal edge and every alternative pixel is illuminated. To avoid this effect, horizontal edges are not considered for processing.
- Another problem with this approach is a sliver. When the angle between adjacent edges is very small, consecutive scan lines produce the same span, which results in aliasing effect. This is called a sliver. Fig. 3.7.2 demonstrates sliver.
- Scan line approach takes advantage of edge coherence. Most of the time, edges intersected by scan line k are also intersected by scan line $k + 1$. If the current intersection point is (x_k, y_k) , the intersection point on the next scan line is computed as,

$$y_{k+1} = y_k + 1$$

$$x_{k+1} = x_k + \frac{1}{m}$$

- This computation is identical to the line drawing algorithm.
- This approach uses two data structures, called Edge Table (ET) and Active Edge Table (AET). Edge table maintains the following information for each edge in a polygon, except horizontal edges.

Y_{MAX}	X_{MIN}	$\frac{1}{m}$	*ptr
-----------	-----------	---------------	------

- Bucket sort algorithm is used to build an edge table. Each entry in the edge table maintains an edge list in increasing order of their x-coordinate. Each bucket holds the above mentioned three values of a particular edge.

3.7.2 Algorithm

- After setting up the edge table, scan line algorithm works as follow :
1. Set Y_{SCAN} to smallest y for which there is an entry in ET.
 2. AET is initially empty.
 3. Repeat following steps until both, AET and ET are empty.
 - (i) Move edges from ET to AET whose $y_{min} = Y_{SCAN}$.
 - (ii) Fill the visible span.
 - (iii) Remove edges with $y_{max} = Y_{SCAN}$.

(iv) Increment y by 1.

(v) For each edges in AET, update x-coordinate.

(vi) Sort all edges on x.

This algorithm optimizes the computation by using edge coherence to compute x and scan line coherence to compute visible span. Sorting in step 3(vi) operates on a small number of edges and on the almost sorted list, so bubble sort or insertion sort would be a good choice to sort them in roughly $O(N)$ time.

3.7.3 Pros and Cons

Advantages

- It takes the advantage of edge coherence, which results in faster execution.
- It only draws visible pixels.
- It requires less memory.
- This is a common choice in a software routine.

Disadvantages

- It is more complex.
- It requires all polygon should be sent to render before drawing.

3.7.4 Example

Ex. 3.7.1 : Determine the visible span for each scan line for given polygon using scan line polygon filling algorithm.

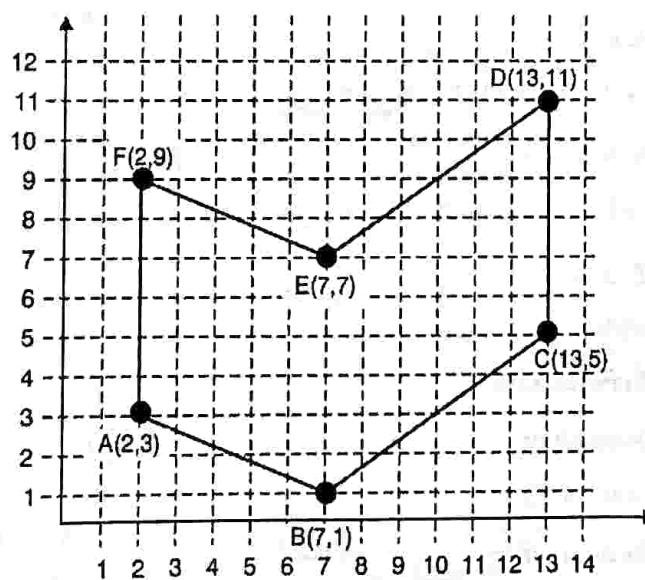


Fig. P. 3.7.1



Soln. :

Edge table for a given polygon is shown in Fig. P. 3.7.1(a).

9	λ												
8	λ												
7	•	→	9	7	-5/2	•	→	11	7	3/2	■		
6	λ												
5	•	→	11	13	0	■							
4	λ												
3	•	→	9	2	0	■							
2	λ												
1	•	→	3	7	-5/2	■	→	5	7	3/2	■		
0	λ												
			Y_{\max}	X_{\min}	$1/m$			Y_{\max}	X_{\min}	$1/m$			

Fig. P. 3.7.1(a)

Let us compute the visible span for each scan line.

Scan line $Y = 1$: AB and BC edges are added

Edge AB : $Y_{\text{scan}} < Y_{\max(\text{AB})}$, So continue

$$X_{\text{AB}} = 7, \text{ So } (X_{\text{AB}}, Y_{\text{SCAN}}) = (7, 1)$$

Edge BC : $Y_{\text{scan}} < Y_{\max(\text{BC})}$, So continue

$$X_{\text{BC}} = 7, \text{ So } (X_{\text{BC}}, Y_{\text{SCAN}}) = (7, 1)$$

Draw pixels from (7, 1) to (7, 1)

Scan line $Y = 2$: No new Edges are added

Edge AB : $Y_{\text{scan}} < Y_{\max(\text{AB})}$, So continue

$$X_{\text{AB}} = X_{\text{AB}} + (1/m) = 7 - 2.5 = 4.5, \text{ So } (X_{\text{AB}}, Y_{\text{SCAN}}) = (5, 2)$$

Edge BC : $Y_{\text{scan}} < Y_{\max(\text{AB})}$, So continue

$$X_{\text{BC}} = X_{\text{BC}} + (1/m) = 7 + 1.5 = 8.5, \text{ So } (X_{\text{BC}}, Y_{\text{SCAN}}) = (8, 2)$$

Draw pixels from (5, 2) to (8, 2)

Scan line $Y = 3$: AF Edge is added

Edge AB : $Y_{\text{scan}} \geq Y_{\max(\text{AB})}$, So remove AB

Edge BC : $Y_{\text{scan}} < Y_{\max(\text{BC})}$, So continue

$$X_{\text{BC}} = X_{\text{BC}} + (1/m) = 8.5 + 1.5 = 10, \text{ So } (X_{\text{BC}}, Y_{\text{SCAN}}) = (10, 3)$$

Edge AF : $Y_{\text{scan}} < Y_{\max(\text{AF})}$, So continue

$$X_{\text{AF}} = 2, \text{ So } (X_{\text{AF}}, Y_{\text{SCAN}}) = (2, 3)$$

Draw pixels from (2, 3) to (10, 3)

Scan line Y = 4 : No New Edges are added

Edge BC : $Y_{\text{scan}} < Y_{\text{max(BC)}}$, So continue

$$X_{BC} = X_{BC} + (1/m) = 10 + 1.5 = 11.5, \text{ So } (X_{BC}, Y_{\text{SCAN}}) = (11, 4)$$

Edge AF : $Y_{\text{scan}} < Y_{\text{max(AF)}}$, So continue

$$X_{AF} = X_{AF} + (1/m) = 2 + 0 = 2, \text{ So } (X_{AF}, Y_{\text{SCAN}}) = (2, 4)$$

Draw pixels from (2, 4) to (11, 4)

Scan line Y = 5 : CD Edge is added

Edge BC : $Y_{\text{scan}} \geq Y_{\text{max(BC)}}$, So remove BC

Edge AF : $Y_{\text{scan}} < Y_{\text{max(AF)}}$, So continue

$$X_{AF} = X_{AF} + (1/m) = 2 + 0 = 2, \text{ So } (X_{AF}, Y_{\text{SCAN}}) = (2, 5)$$

Edge CD : $Y_{\text{scan}} < Y_{\text{max(CD)}}$, So continue

$$X_{CD} = 13, \text{ So } (X_{CD}, Y_{\text{SCAN}}) = (13, 5)$$

Draw pixels from (2, 5) to (13, 5)

Scan line Y = 6 : No New Edges are added

Edge AF : $Y_{\text{scan}} < Y_{\text{max(AF)}}$, So continue

$$X_{AF} = X_{AF} + (1/m) = 2 + 0 = 2, \text{ So } (X_{AF}, Y_{\text{SCAN}}) = (2, 6)$$

Edge CD : $Y_{\text{scan}} < Y_{\text{max(CD)}}$, So continue

$$X_{CD} = X_{CD} + (1/m) = 13 + 0 = 13, \text{ So } (X_{CD}, Y_{\text{SCAN}}) = (13, 6)$$

Draw pixels from (2, 6) to (13, 6)

Scan line Y = 7 : EF and DE Edges are added

Edge AF : $Y_{\text{scan}} < Y_{\text{max(AF)}}$, So continue

$$X_{AF} = X_{AF} + (1/m) = 2 + 0 = 2, \text{ So } (X_{AF}, Y_{\text{SCAN}}) = (2, 7)$$

Edge CD : $Y_{\text{scan}} < Y_{\text{max(CD)}}$, So continue

$$X_{CD} = X_{CD} + (1/m) = 13 + 0 = 13, \text{ So } (X_{CD}, Y_{\text{SCAN}}) = (13, 7)$$

EF and DE Edges are added.

Edge EF : $Y_{\text{scan}} < Y_{\text{max(EF)}}$, So continue

$$X_{EF} = 7, \text{ So } (X_{EF}, Y_{\text{SCAN}}) = (7, 7)$$

Edge DE : $Y_{\text{scan}} < Y_{\text{max(DE)}}$, So continue

$$X_{DE} = 7, \text{ So } (X_{DE}, Y_{\text{SCAN}}) = (7, 7)$$

Draw pixels from (2, 7) to (7, 7) and (7, 7) to (13, 7)

Scan line Y = 8 : No New Edges are added

Edge AF : $Y_{\text{scan}} < Y_{\text{max(AF)}}$, So continue



$$X_{AF} = X_{AF} + (1/m) = 2 + 0 = 2, \text{ So } (X_{AF}, Y_{SCAN}) = (2, 8)$$

Edge CD : $Y_{SCAN} < Y_{max(CD)}$, So continue

$$X_{CD} = X_{CD} + (1/m) = 13 + 0 = 13, \text{ So } (X_{CD}, Y_{SCAN}) = (13, 8)$$

Edge EF : $Y_{SCAN} < Y_{max(EF)}$, So continue

$$X_{EF} = X_{EF} + (1/m) = 7 - 2.5 = 4.5, \text{ So } (X_{EF}, Y_{SCAN}) = (4, 8)$$

Edge DE : $Y_{SCAN} < Y_{max(DE)}$, So continue

$$X_{DE} = X_{DE} + (1/m) = 7 + 1.5 = 8.5, \text{ So } (X_{DE}, Y_{SCAN}) = (9, 8)$$

Draw pixels from (2, 8) to (4, 8) and (9, 8) to (13, 8)

Scan line $Y = 9$: No New Edges are added

Edge AF : $Y_{SCAN} \geq Y_{max(AF)}$, So Remove AF

Edge CD : $Y_{SCAN} < Y_{max(CD)}$, So continue

$$X_{CD} = X_{CD} + (1/m) = 13 + 0 = 13, \text{ So } (X_{CD}, Y_{SCAN}) = (13, 9)$$

Edge EF : $Y_{SCAN} \geq Y_{max(EF)}$, So remove EF

Edge DE : $Y_{SCAN} < Y_{max(DE)}$, So continue

$$X_{DE} = X_{DE} + (1/m) = 8.5 + 1.5 = 10, \text{ So } (X_{DE}, Y_{SCAN}) = (10, 9)$$

Draw pixels from (10, 9) to (13, 9)

Scan line $Y = 10$: No New Edges are added

Edge CD : $Y_{SCAN} < Y_{max(CD)}$, So continue

$$X_{CD} = X_{CD} + (1/m) = 13 + 0 = 13, \text{ So } (X_{CD}, Y_{SCAN}) = (13, 10)$$

Edge DE : $Y_{SCAN} < Y_{max(DE)}$, So continue

$$X_{DE} = X_{DE} + (1/m) = 10 + 1.5 = 11.5, \text{ So } (X_{DE}, Y_{SCAN}) = (12, 10)$$

Draw pixels from (12, 10) to (13, 10)

Scan line $Y = 11$: No New Edges are added

Edge CD : $Y_{SCAN} \geq Y_{max(CD)}$, So remove CD

Edge DE : $Y_{SCAN} < Y_{max(DE)}$, So remove DE

AET is empty, so STOP

Scan Line	Visible Span
$Y_{SCAN} = 1$	(7, 1) to (7, 1)
$Y_{SCAN} = 2$	(5, 2) to (8, 2)
$Y_{SCAN} = 3$	(2, 3) to (10, 3)
$Y_{SCAN} = 4$	(2, 4) to (11, 4)
$Y_{SCAN} = 5$	(2, 5) to (13, 5)

Scan Line	Visible Span
$Y_{SCAN} = 6$	(2, 6) to (13, 6)
$Y_{SCAN} = 7$	(2, 7) to (7, 7) and (7, 7) to (13, 7)
$Y_{SCAN} = 8$	(2, 8) to (4, 8) and (9, 8) to (13, 8)
$Y_{SCAN} = 9$	(10, 9) to (13, 9)
$Y_{SCAN} = 10$	(12, 10) to (13, 10)

3.8 Seed Fill vs Scan Line Algorithm

Sr. No.	Seed Fill Algorithm	Scan Line Algorithm
1.	Simple to implement.	Very complex.
2.	Operates in screen space.	Operates in screen and / or object space.
3.	Require system call to get pixel intensity.	It is device-independent.
4.	Need to specify seed point contained within the polygon.	No need to specify seed point.
5.	Need extensive stacking.	No need for stacking.
6.	Generally used in graphics packages.	Typically used in interactive rendering.
7.	Not suitable for Z-Buffer.	Suitable for Z-Buffer.

Review Questions

- Q. 1 List types of Polygon.
- Q. 2 What is a polygon? What are the different types of polygons?
- Q. 3 Explain two methods for testing whether the point is inside the polygon or not.
- Q. 4 Explain the different methods for testing a pixel inside a polygon.
- Q. 5 Describe the Even-Odd test method.
- Q. 6 Explain the winding number method.
- Q. 7 List various polygon filling algorithms.
- Q. 8 Explain the boundary fill algorithm with pseudo-code. Also mention its limitations, if any.
- Q. 9 Write an algorithm for boundary fill.
- Q. 10 State merits and demerits of boundary fill algorithm.
- Q. 11 Write algorithm to fill the polygon area using flood fill method.
- Q. 12 Write a short note on polygon filling with patterns.



Two Dimensional Geometric Transformations

Module 3

Syllabus

Basic transformations : Translation, Scaling, Rotation, Matrix representation and Homogeneous Coordinates,
Composite transformation, Other transformations : Reflection and Shear

4.1 Introduction

- In the real world, the user frequently rearranges objects. Picture or scene in a computer is also made up of objects. A graphics system should allow users to change the way by which the object appears.
- The process of changing the scale, shape or position of the object is called **transformation**.

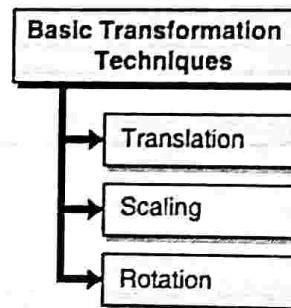


Fig. 4.1.1 : Basic transformation techniques

- Transformation is a very common operation used in graphics applications. Translation, rotation and scaling are the basic transformation operations. Reflection and shearing are other common transformation operations.
- Translation, rotation and reflection are also known as **rigid body transformations** because they do not alter the shape of the object. They just reposition the object from one location to another location.
- Scaling and shearing deform the shape of the object under consideration, so they are not rigid-body transformation.
- The main objective of two-dimensional transformations is to simulate movement and manipulate 2D objects in the XY plane.

Terminology

The following terminology will be used for the rest of the discussion.

P = Set of points of the object which has to be transformed.

P' = Set of points of the object which has been transformed.

M = Transformation matrix.

Transformation is a function of object coordinates with various operations as another argument.

$$\therefore P' = f(P, M)$$

Where, M is a transformation matrix to be applied on a set of points P . More common representation of transformation operation is $P' = M \cdot P$.

Translation, rotation and reflection are also known as **rigid body transformations** because they do not alter the shape of the object. They just reposition the object from one location to another location. Scaling and shearing deform the shape of the object under consideration, so they are not rigid-body transformation.

Assumption : Matrices are represented in column-major form. For row-major representation, $P' = P \cdot M$

4.2 Matrix Operations

Transformation and matrix operations go hand in hand. Let us first discuss various representation, properties and operations on matrix.

4.2.1 Matrix Representation

The matrix can be represented in two ways.

1. Column Measure Representation

In this representation, point coordinates are represented in the column.

For example,

$$A = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Let $M = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$ be some transformation matrix. The coordinates of transformed coordinates are given by,

$$P' = M \cdot P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\therefore x' = ax + cy$$

$$y' = bx + dy$$

In column major representation, M is pre-multiplied to original points of an object.

2. Row Measure Representation

- In this representation, point coordinates are represented in row. For example,

$$B = A^T = [x \ y] = [2 \ 3]$$

$$P'^T = P^T \cdot M^T$$

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\therefore x' = ax + cy$$



$$y' = bx + dy$$

- In row-major representation, M is post-multiplied to original points of an object.
- We can multiply two matrices A and B only if dimensions of A are $p \times q$ and dimensions of B are $q \times r$. For matrix multiplication, it is necessary to have a number of columns in the first matrix equivalent to the number of rows in the second matrix.

$$A_{p \times q} \times B_{q \times r} = C_{p \times r}$$

- If number of columns in A and number of rows in B are not same, then matrix dimensions are not compatible and multiplication is not possible.

4.2.2 Matrix Properties

1. **Associative** : Matrix multiplication is associative. i.e.,

$$A \cdot B \cdot C = A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

2. **Commutative** : Matrix multiplication is not commutative. i.e.,

$$A \cdot B \neq B \cdot A$$

4.2.3 Determinant of Matrix

Determinant of 2×2 matrix is calculated as below :

$$\text{Let } A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$\text{Determinant } \Delta(A) = |A| = a_{11}a_{22} - a_{12}a_{21}$$

Determinant of 3×3 matrix is calculated as below :

$$\text{Let } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\text{Determinant } \Delta(B) = |B| = b_{11}(b_{22}b_{33} - b_{23}b_{32}) - b_{12}(b_{21}b_{33} - b_{23}b_{31}) + b_{13}(b_{21}b_{32} - b_{22}b_{31})$$

4.2.4 Multiplying Two Matrices

Let A and B are 2×2 square matrices to be multiplied and C is the resultant matrix.

$$\text{If, } A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ and } B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C = A \times B$$

$$\therefore \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\text{Where, } C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

For some random dimensional compatible, rectangular matrices A and B,

$$C = A \times B$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix}$$

$$\text{Where, } C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} + A_{23} \cdot B_{31}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22} + A_{23} \cdot B_{32}$$

4.3 Translation

- The translation is defined as a shifting of an object along a straight path.
- Translation does not alter the shape or size of the object. It just moves the entire object from one location to another location along a straight path.
- This is achieved by first shifting object in the X-direction and then in the Y-direction by amount t_x and t_y , respectively.
- The vector $T = [t_x \ t_y]$, specifying the amount of shift in both the directions is called **shift-vector** or **translation vector**.
- Translation operation is formulated as,

$$x' = x + t_x$$

$$y' = y + t_y$$

Where (x, y) is the original point, $[t_x, t_y]$ is shift vector and (x', y') is translated point.

Equivalent matrix representation of this operation is,

$$P' = T + P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

- If an object has more than one vertex, we can apply a transformation to individual vertex and then join them all to reconstruct the object.
- Rigid body transformation moves object without deformation. Amount of shift is identical for all the points.
- In Fig. 4.3.1, object O is translated by shift-vector $[6, 5]$. O' is the translated object.

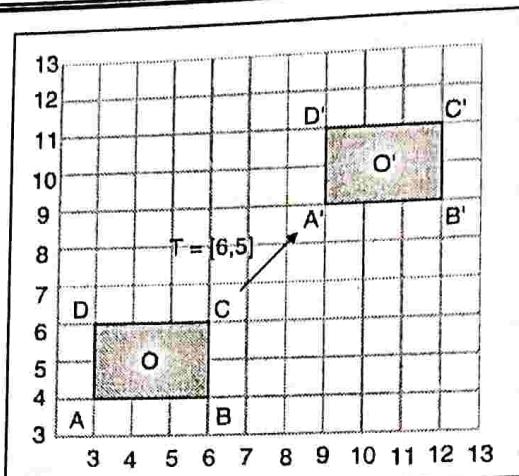


Fig. 4.3.1 : Translation operation

Ex. 4.3.1 : Translate a triangle with coordinates A(2,2), B(6,2) and C(4,4) with translation vector $[4 \ 7]^T$.

Soln. :

$$\text{Here translation vector, } T = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

Translation operation is defined as,

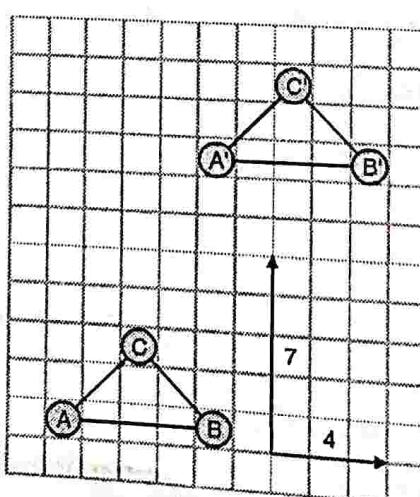
$$P' = T + P$$

$$\therefore A' = T + A = \begin{bmatrix} 4 \\ 7 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \end{bmatrix}$$

$$B' = T + B = \begin{bmatrix} 4 \\ 7 \end{bmatrix} + \begin{bmatrix} 6 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 9 \end{bmatrix}$$

$$C' = T + C = \begin{bmatrix} 4 \\ 7 \end{bmatrix} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 8 \\ 11 \end{bmatrix}$$

After translation, triangle A(2,2), B(6,2), C(4,4) will move to A'(6,9), B'(10,9), C'(8,11).

Fig. P. 4.3.1 : Translation of triangle ABC by shift-vector $[4 \ 7]^T$

Original Coordinates	Transformed Coordinates
A(2, 2)	A' (6, 9)
B(6, 2)	B' (10, 9)
C(4, 4)	C' (8, 11)

Ex. 4.3.2 : Explain the translation of a line in detail.

Soln. :

- The translation is a rigid-body transformation.
- After translation operation, the shape of the object does not change. That gives the intuition to translate only endpoints of line and again joining them.
- We do not need to translate individual pixel on line. Let us consider the line with endpoints A (x_1, y_1) and B (x_2, y_2). Let's assume the translation vector is $T [t_x \ t_y]^T$.
- Translation of line requires translation of only endpoints of the line. Translated endpoints are further passed to line generation algorithm to generate remaining pixels on the straight path joining them.

If (x'_1, y'_1) and (x'_2, y'_2) are the translated coordinates, then

$$x'_1 = t_x + x_1$$

$$y'_1 = t_y + y_1$$

$$x'_2 = t_x + x_2$$

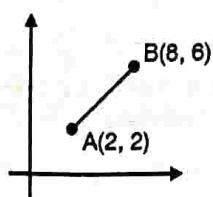
$$y'_2 = t_y + y_2$$

In matrix form,

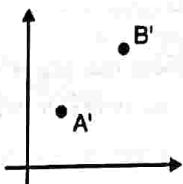
$$P' = T \cdot P$$

Where, P' and P are set of translated and original points respectively. Homogeneous representation (Discussed in section 4.6) of transformation sequence would be,

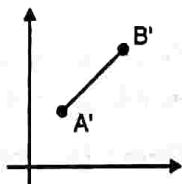
$$\therefore P' = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \\ 1 & 1 \end{bmatrix}$$



(a) Original line



(b) Translated endpoints



(c) Points on the line after joining endpoint A' B'

Fig. P. 4.3.2 : Translation of line



4.4 Rotation

- 2D rotation is described by repositioning all the points of an object along a circular path in the 2D plane.
- To perform rotation, we need a rotation angle θ and the reference point (x_r, y_r) with respect to which object is to be rotated. The reference point is also called a pivot point.
- If rotation is performed in an anticlockwise direction, the value of the angle is considered positive, otherwise it is negative. Rotation directions are shown in Fig. 4.4.1.

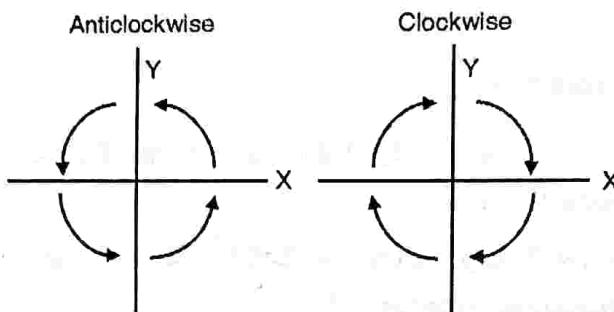


Fig. 4.4.1 : Anticlockwise and clockwise rotation direction

- Rotation can be performed with respect to origin or with respect to some reference point. We will discuss both the cases here.

4.4.1 Rotation about Origin

- Let's derive the transformation matrix for rotation about the origin. As shown in Fig. 4.4.2, $P(x, y)$ is the original point, which is to be rotated in XY plane by the angle θ in an anticlockwise direction. $P'(x', y')$ is the rotated point.

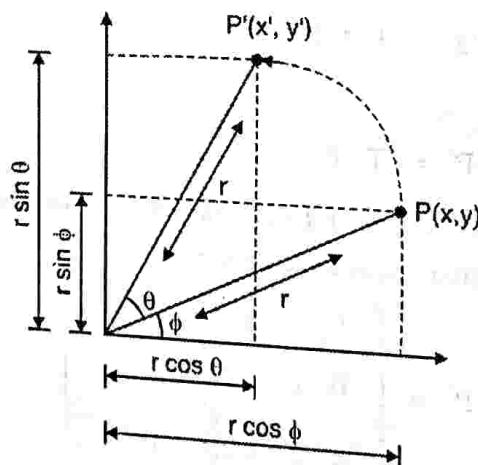


Fig. 4.4.2 : Rotational geometry of point P

- The distance of P from the origin is let's say r . Consider the angle of P with X-axis is ϕ and is rotated by angle θ in an anticlockwise direction, so point P' makes an angle $(\phi + \theta)$ with X-axis.
- From Fig. 4.4.2, Using trigonometric rule, for Point P,

$$\sin \phi = \frac{y}{r}$$

$$\cos \phi = \frac{x}{r}$$

$$\therefore x = r \cos \phi \quad \dots(4.4.1)$$

$$y = r \sin \phi \quad \dots(4.4.2)$$

For point P',

$$\cos(\phi + \theta) = \frac{x'}{r}$$

$$\sin(\phi + \theta) = \frac{y'}{r}$$

$$\text{So, } x' = r \cos(\phi + \theta) = r \cos \phi \cdot \cos \theta - r \sin \phi \cdot \sin \theta \quad \dots(4.4.3)$$

$$y' = r \sin(\phi + \theta) = r \cos \phi \cdot \sin \theta + r \sin \phi \cdot \cos \theta \quad \dots(4.4.4)$$

By putting value of Equations (4.4.1) and (4.4.2) in Equation (4.4.3) and Equation (4.4.4), we get,

$$x' = x \cos \theta - y \sin \theta \quad \dots(4.4.5)$$

$$y' = x \sin \theta + y \cos \theta \quad \dots(4.4.6)$$

Matrix representation of above equation is written as;

$$P' = R \cdot P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \dots(4.4.7)$$

Where, $R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ represents rotation matrix for rotation about origin

Inverse rotation is defined by considering negative value of θ

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

The inverse rotation matrix is equal to the transpose of the original matrix. Such a matrix is called the **orthogonal matrix**.

Properties of the Pure Rotation Matrix

- The determinant is always 1.
- The matrix should be orthogonal.

Note: 1. If the reference point is not specified, rotation is by default about the origin.

2. If rotation direction is not specified, by default it is in anticlockwise.

4.4.2 Rotation with Respect to Reference Point

Q. Derive the matrix for 2D rotation about an arbitrary point.

MU - Dec. 18, 10 Marks

Q. Explain the steps for 2D rotation about an arbitrary point.

MU - May 19, 10 Marks



Matrix sequence of Equation (4.4.7) is only valid for rotation about the origin. If we want to rotate any point or object with respect to some reference point (x_r, y_r) , then we should perform the following steps.

1. Translate the reference point (x_r, y_r) to origin : Translation vector is given as,

$$T = \begin{bmatrix} -x_r \\ -y_r \end{bmatrix}$$

2. Apply rotation by given angle θ . The rotation matrix is given as,

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

3. Translate reference point back to its actual location.

$$T^{-1} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

The sequence of operations is shown in Fig. 4.4.3.

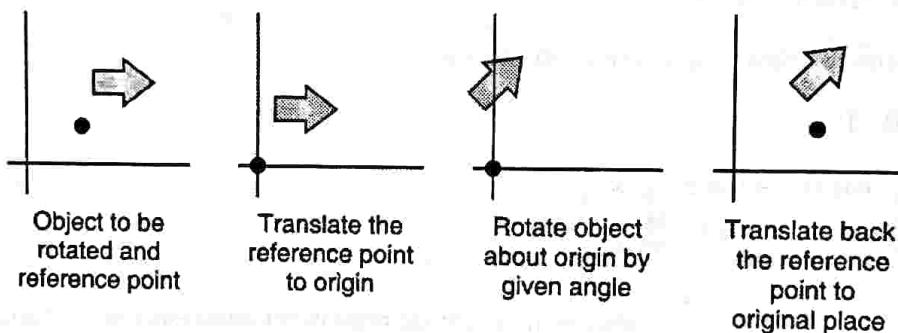


Fig. 4.4.3 : Transformation sequence for rotation with respect to the reference point

For column major representation, the sequence of operations is defined as :

$$P' = [T^{-1} + [R[T + P]]]$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \left\{ \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \left\{ \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \left\{ \begin{bmatrix} -x_r \\ -y_r \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \right\} \right\}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \left\{ \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \left\{ \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} \right\} \right\}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \begin{bmatrix} (x - x_r) \cos \theta - (y - y_r) \sin \theta \\ (x - x_r) \sin \theta + (y - y_r) \cos \theta \end{bmatrix}$$

So, $x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta$

$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$

Note : For row-major representation, the sequence of operations would be: $P' = [(P + T[R]) + T^{-1}]$

Ex. 4.4.1 : Rotate a point P(3, 2) around the origin in a counter-clockwise direction by 90°.

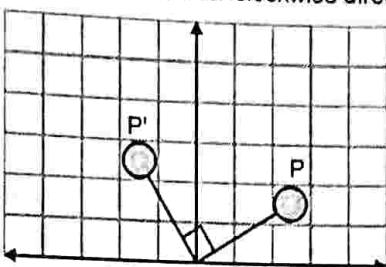


Fig. P. 4.4.1 : Rotation of point P(2, 3) by 90° in the counter-clockwise direction

Soln. :

Here, the direction of rotation is anticlockwise, So, $\theta = +90^\circ$

$$\therefore P' = R \cdot P = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

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

Thus, point P(3, 2) becomes P'(-2, 3) after rotation by 90° in the anti-clockwise direction.

Original Coordinates	Transformed Coordinates
A(3, 2)	A'(-2, 3)

Ex. 4.4.2 : Rotate a point A(3, 2) by 90° in anticlockwise direction with respect to reference point B(1, 2).

Soln. :

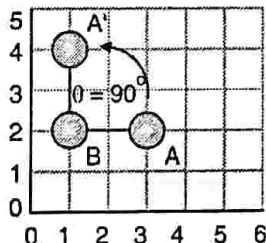


Fig. P. 4.4.2 : Output of said rotation operation

Here, Rotation direction is anticlockwise, so $\theta = +90^\circ$

We have to rotate a point around the reference point

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Rotation about reference point (x_r, y_r) is given by,

$$\therefore x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta$$

$$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$$

$$x' = 1 + (3 - 1) \cos 90^\circ - (2 - 2) \sin 90^\circ = 1 + (2) 0 - (0) 1$$

$$\therefore x' = 1$$



$$y' = 2 + (3 - 1) \sin 90^\circ + (2 - 2) \cos 90^\circ = 2 + (2) 1 + (0) 0$$

$$\therefore y' = 4$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

Thus, point A(3, 2) becomes A'(1, 4) after rotation by 90° in anticlockwise direction with respect to the reference point B(1, 2).

Original Coordinates	Transformed Coordinates
A(3, 2)	A' (1, 4)

Ex. 4.4.3 : Consider a square P(0, 0), Q(0, 10), R(10, 10), S(10, 0). Rotate the square anticlockwise about fixed point R(10, 10) by an angle 45 degree.

Soln. :

Here, given the square P(0, 0), Q(0, 10), R(10, 10), S(10, 0).

Square is to be rotated about a reference point R(10, 10) by 45° . The desired operation is achieved by performing the following steps :

1. Translate reference point R to the origin.
2. Perform rotation by 45° in an anticlockwise direction.
3. Inverse translation of point R.

The transformation matrix for this sequence is given by $M = T^{-1} \cdot R_{(\theta = 45^\circ)} \cdot T$

$$P' = M \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45^\circ) & -\sin(45^\circ) & 0 \\ \sin(45^\circ) & \cos(45^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 10 & 10 \\ 0 & 10 & 10 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.709 & -0.709 & 0 \\ 0.709 & 0.709 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 10 & 10 \\ 0 & 10 & 10 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply first two and last two matrices,

$$= \begin{bmatrix} 0.709 & -0.709 & 10 \\ 0.709 & 0.709 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -10 & -10 & 0 & 0 \\ -10 & 0 & 0 & -10 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 10 & 2.91 & 10 & 17.09 \\ -4.18 & 2.91 & 10 & 2.91 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Table P. 4.4.3

Original Coordinates	Transformed Coordinates
P (0, 0)	P' (10, -4.18)
Q (0, 10)	Q' (2.91, 2.91)
R (10, 10)	R' (10, 10)
S (10, 0)	S' (17.09, 2.91)

4.5 Scaling

Q. Explain the steps for 2D rotation about arbitrary point and provide a composite transformation for the same.

MU - May 18, 10 Marks

- Translation and rotation change the position of an object; they do not deform the shape. Scaling may alter the shape of an object.
- Scaling can be performed with respect to origin or some reference point.

4.5.1 Scaling with Respect to Origin

Scaling is performed by multiplying the vertex coordinates by scaling parameters S_x and S_y respectively.

So,

$$x' = S_x \cdot x$$

$$y' = S_y \cdot y$$

We can represent the scaling operation in matrix form as,

$$P' = S \cdot P$$

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

S_x, S_y are any positive numeric values.

Note: For 2×2 matrix representation, translation is an additive operation, whereas scaling and rotation are multiplicative.

When the object is scaled up with respect to the origin, it goes away from the origin. The converse is true when the object is scaled down. Scaling may be in one direction or both directions, or even it can be uniform ($S_x = S_y$) or non-uniform ($S_x \neq S_y$).

Various Cases of Scaling

$(S_x = S_y) = 1 \Rightarrow$ No scaling

$(S_x = S_y) \neq 1 \Rightarrow$ Uniform scaling

$S_x \neq S_y \Rightarrow$ Non-uniform scaling

$(S_x = S_y) < 1 \Rightarrow$ Uniform compression



$(S_x = S_y) > 1 \Rightarrow$ Uniform expansion

$S_x > 1 \Rightarrow$ X-coordinates of the object will move away from the origin

$S_x < 1 \Rightarrow$ X-coordinates of the object will move near to the origin

$S_y > 1 \Rightarrow$ Y-coordinates of the object will move away from the origin

$S_y < 1 \Rightarrow$ Y-coordinates of the object will move near to the origin

As shown in Fig. 4.5.1, line AB has coordinates A(1, 1) and B(1, 4). With $S_x = 3$, $S_y = 2$, the line goes away from the origin and we get A'(3, 2) and B'(3, 8).

$$A' = S \cdot A = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$B' = S \cdot B = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

Similarly, if we select scaling factor less than one, the object comes closer to the origin. With $S_x = S_y = 0.5$, line CD with endpoints C(10, 4) and D(10, 8) comes at C'(5, 2) and D'(5, 4).

$$C' = S \cdot C = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 10 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$D' = S \cdot D = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 10 \\ 8 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

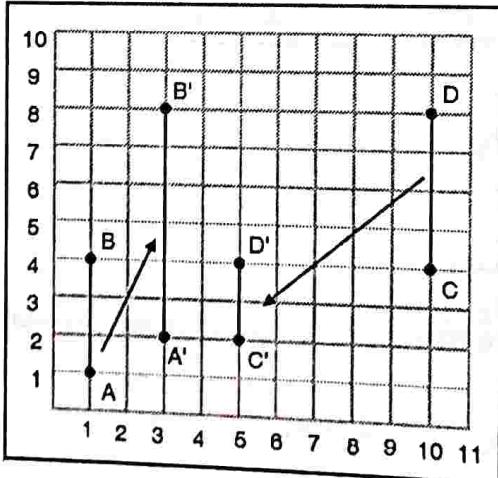


Fig. 4.5.1 : Demonstration of scaling

Note : These scaling formulas are only applicable if scaling is with respect to the origin.

4.5.2 Scaling with Respect to Reference Point

- In this section, we will derive the transformation matrix to scale the object with respect to some reference point.
- Let (x_r, y_r) be the reference point, with respect to which object is to be scaled.

Following steps should be performed to carry out the desired operation.

1. Translate the reference point to the origin.

$$T = \begin{bmatrix} -x_r \\ -y_r \end{bmatrix}$$

2. Apply scaling on the translated object.

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

3. Translate reference point back to its actual location.

$$T^{-1} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

The sequence of operations is shown in Fig. 4.5.2.

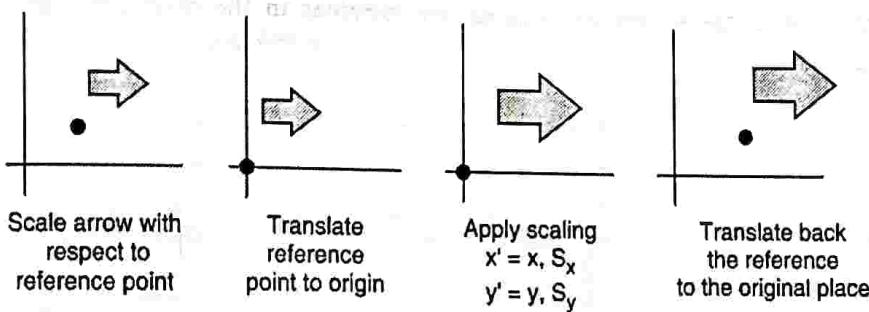


Fig. 4.5.2 : Sequence of transformation operations for scaling with respect to a reference point

- Recall that the transformation sequence is written from right to left for column-major representation. So, the transformed coordinates of an object are given by,

$$P' = [T^{-1} + [S[T + P]]]$$

$$P' = \left\{ \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \left\{ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \left\{ \begin{bmatrix} -x_r \\ -y_r \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \right\} \right\}$$

$$= \left\{ \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \left\{ \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} \right\} \right\} = \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \begin{bmatrix} (x - x_r)S_x \\ (y - y_r)S_y \end{bmatrix}$$

$$x' = x_r + (x - x_r)S_x = S_x \cdot x + x_r \cdot (1 - S_x)$$

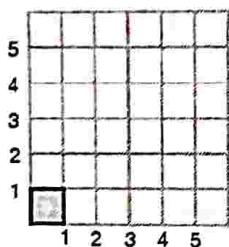
$$y' = y_r + (y - y_r)S_y = S_y \cdot y + y_r \cdot (1 - S_y)$$

- As scaling parameters and reference points are fixed, the terms $(x_r \cdot (1 - S_x))$ and $(y_r \cdot (1 - S_y))$ are constant for all the vertices of the object.



4.5.3 Uniform vs. Non-Uniform Scaling

- If scaling parameters are same for both the directions, it is called **uniform scaling**, otherwise, it is called **non-uniform scaling**.
- Uniform scaling only scale up or scale down the object size; it does not change the shape of the object. In the case of uniform scaling, the square remains square and circle remains circle after scaling operation, only the size of an object alters.
- Non-uniform scaling may alter the shape of the object. In the case of non-uniform scaling, the circle may turn out to be an ellipse, or square may turn out to be a rectangle.
- Consider the unit square with vertices A(0, 0), B(1, 0), C(1, 1) and D(0, 1). If we apply uniform scaling on this unit square with $S_x = S_y = 2$, we get A'(0, 0), B'(2, 0), C'(2, 2) and D'(0, 2). And if we apply non uniform scaling with $S_x = 2$ and $S_y = 3$, we get A''(0, 0), B''(2, 0), C''(2, 3) and D''(0, 3). Fig. 4.5.3 illustrates the concept of uniform and non-uniform scaling.
- In case of uniform scaling, the square remains square, whereas in the case of non-uniform scaling square turns into a rectangle.



(a) Unit square

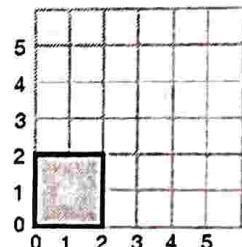
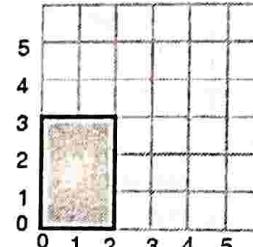
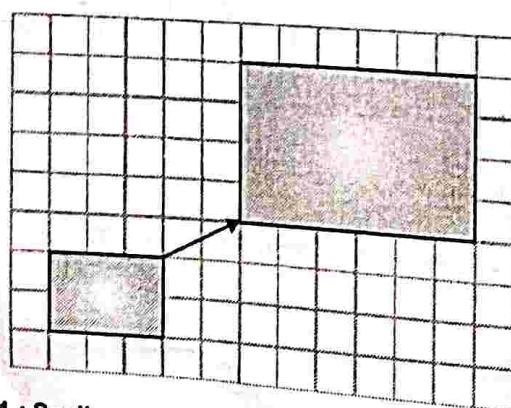
(b) Uniform scaling with
 $S_x = S_y = 2$ (c) Non uniform scaling with
 $S_x = 2, S_y = 3$

Fig. 4.5.3 : Uniform and non-uniform scaling

- Scaling is called differential scaling if $S_x \neq S_y$. Differential scaling is used to construct complex shapes with the help of basic translation and rotation transformation. Differential scaling is also known as non-uniform scaling.

Ex. 4.5.1 : Scale a square with coordinates A(2,2), B(4,2), C(4,4), D(2,4), Where; $S_x = 3$ and $S_y = 2$.

Soln. :

Fig. P. 4.5.1 : Scaling of object by scaling parameters, $S = [3 \ 2]^T$

Transformation sequence for scaling is defined by,

$$P' = S \cdot P = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\therefore A' = S \cdot A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

$$B' = S \cdot B = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 4 \end{bmatrix}$$

$$C' = S \cdot C = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix}$$

$$D' = S \cdot D = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A(2, 2)	A'(6, 4)
B(4, 2)	B'(12, 4)
C(4, 4)	C'(12, 8)
D(2, 4)	D'(6, 8)

Ex. 4.5.2: Scale a triangle with vertices A(2,2), B(6,2) and C(4,4) with respect to a reference point (3,4) with $S_x = 2$ and $S_y = 3$.

Soln. :

Here, Scaling is with respect to a reference point. We have to first translate reference point to origin then apply scaling and then translate the reference point back to the actual location. Transformed coordinates of this sequence of operation are given by,

$$x' = S_x \cdot x + x_r(1 - S_x)$$

$$y' = S_y \cdot y + y_r(1 - S_y)$$

$x_r(1 - S_x)$ and $y_r(1 - S_y)$ are constant for all points.

$$x_r(1 - S_x) = 3(1 - 2) = -3$$

$$y_r(1 - S_y) = 4(1 - 3) = -8$$

$$A' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$B' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 12 \\ 6 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 9 \\ -2 \end{bmatrix}$$

$$C' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 8 \\ 12 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$



Original Coordinates	Transformed Coordinates
A(2, 2)	A'(1, -2)
B(6, 2)	B'(9, -2)
C(4, 4)	C'(5, 4)

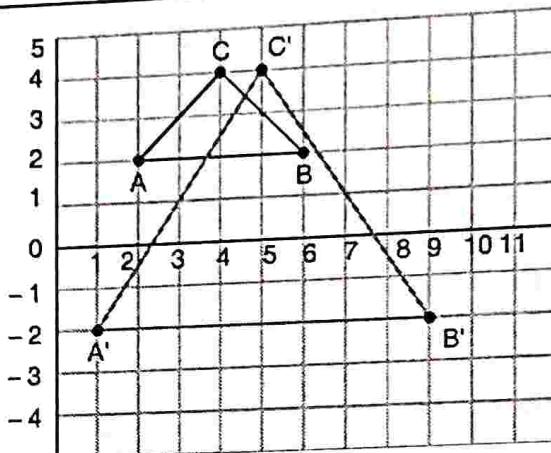


Fig. P. 4.5.2 : Output of scaling operation

Ex. 4.5.3 : Apply scaling to unit square with a bottom left corner at (3, 4) with $S_x = 2$ and $S_y = 0.5$.

Soln. :

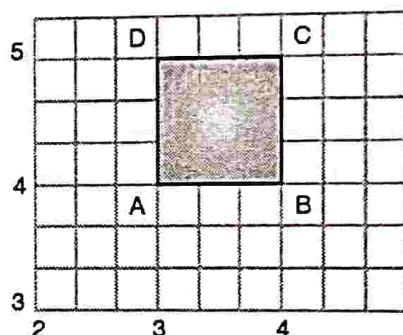


Fig. P. 4.5.3

Assume that A is the corner with coordinates (3, 4). So remaining vertices of the unit square will be B(4, 4), C(4, 5) and D(3, 5).

$$\text{Scaling matrix, } S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

Let us assume P and P' are original and transformed coordinates of an object respectively.

$$\therefore P' = S \cdot P$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 3 & 4 & 4 & 3 \\ 4 & 4 & 5 & 5 \end{bmatrix} = \begin{bmatrix} 6 & 8 & 8 & 6 \\ 2 & 2 & 2.5 & 2.5 \end{bmatrix}$$

Original Coordinate	Transformed Coordinate
A(3, 4)	A' (6, 2)
B(4, 4)	B' (8, 2)
C(4, 5)	C' (8, 2.5)
D(3, 5)	D' (6, 2.5)

Ex. 4.5.4 : Prove that the scaling operation is commutative

Soln. :

- Assume that transformation matrix for scaling S_1 by parameters (S_{x_1}, S_{y_1}) followed by scaling S_2 with parameters (S_{x_2}, S_{y_2}) is $M_1 = S_1 S_2$.

$$\text{L.H.S} = M_1 = S_1 S_2 = \begin{bmatrix} S_{x_1} & 0 \\ 0 & S_{y_1} \end{bmatrix} \begin{bmatrix} S_{x_2} & 0 \\ 0 & S_{y_2} \end{bmatrix} = \begin{bmatrix} S_{x_1} S_{x_2} & 0 \\ 0 & S_{y_1} S_{y_2} \end{bmatrix}$$

- And the transformation matrix for scaling S_2 by parameters (S_{x_2}, S_{y_2}) followed by scaling S_1 with parameters (S_{x_1}, S_{y_1}) is $M_2 = S_2 S_1$.

$$\text{R.H.S} = M_2 = S_2 S_1 = \begin{bmatrix} S_{x_2} & 0 \\ 0 & S_{y_2} \end{bmatrix} \begin{bmatrix} S_{x_1} & 0 \\ 0 & S_{y_1} \end{bmatrix} = \begin{bmatrix} S_{x_1} S_{x_2} & 0 \\ 0 & S_{y_1} S_{y_2} \end{bmatrix} = \text{L.H.S}$$

- Hence the given statement is proved that scaling operation is commutative.

Ex. 4.5.5 : Perform scaling on a triangle (1, 1), (8, 1) and (1, 9) with a scaling factor of 2 in both x and y-directions. Find the final coordinates of the triangle.

Soln. :

Given the triangle A(1, 1), B(8, 1) and C(1, 9) with parameters $S_x = S_y = 2$. In homogenous coordinate representation, the transformed coordinates are given as,

$$P' = S \cdot P$$

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 8 & 1 \\ 1 & 1 & 9 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 16 & 2 \\ 2 & 2 & 18 \\ 1 & 1 & 1 \end{bmatrix}$$

Original Coordinate	Transformed Coordinate
A(1, 1)	A' (2, 2)
B(8, 1)	B' (16, 2)
C(1, 9)	C' (2, 18)



4.6 Matrix Representation and Homogeneous Coordinates

- In real-world applications, it is very common that the object goes through sequences of different geometric transformations.
- The object may require translation, rotation, scaling, shearing or reflection to be applied any number of times in any order.
- In previous sections, we saw that translation operation is additive while scaling and rotation are multiplicative. So all three basic transformations can be represented in generalized form as,

$$P' = M_1 + M_2 P \quad \dots(4.6.1)$$

- For translation, M_2 is the identity matrix (multiplicative identity).
- So, $P' = M_1 + P$ represents the translation matrix
- And for rotation and scaling, M_1 is zero matrix (additive identity),
- So, $P' = M_2 P$, Where M_2 represents rotation or scaling matrix
- Point (x, y) in the Cartesian coordinate has infinite representation in homogeneous coordinates.
- For example, $(x, y) = (x, y, 1) = (2x, 2y, 2) = \dots = (nx, ny, n)$, all correspond to the same representation of (x, y) .

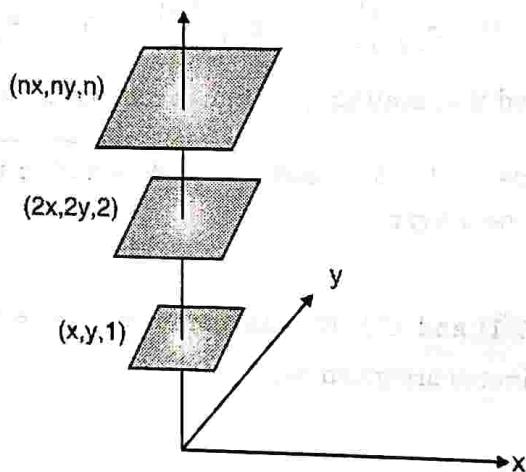


Fig. 4.6.1 : Representation of point in the homogeneous coordinate system

- We will reformulate this matrix representation of Equation (4.6.1) to combine additive and multiplicative matrices in a single matrix by extending 2×2 representation to 3×3 using homogeneous representation.
 - In the homogeneous coordinate system, if we multiply the point by some constant, it represents the same point.
 - Any Cartesian coordinate (x, y) can be represented as (x_h, y_h, h) in a homogeneous system, such that
- $$x = \frac{x_h}{h} \text{ and } y = \frac{y_h}{h}$$
- Homogeneous coordinate introduces the third dimension but it doesn't alter the value of (x, y) .
 - Fig. 4.6.1 explains how we can represent the same points at different levels.

Properties of Homogenous Coordinate Representation

1. Any 2D point in the homogeneous coordinate system is represented by a triplet (x, y, h) , where x, y and h are not all zero. $(0, 0, 0)$ does not represent any point. Origin is represented as $(0, 0, 1)$.
2. Inhomogeneous coordinate systems, two points are identical, if one point is derived by multiplying some constant to the second point.
3. If h is not zero, then point (x_h, y_h, h) in a homogenous coordinate system is represented as $(x_h/h, y_h/h)$ in the Euclidean/Cartesian coordinate system.
4. If h is 0, point represented is at infinity.
 - Homogenous representation allows us to write all geometric transformation equations in matrix multiplication form.
 - It brings uniformity in operation. Implementation of transformation operation in programming language becomes easier with this representation as all the operations are performed using matrix multiplication operations only.
 - Basic transformation operations matrix using homogeneous coordinate are shown below.

4.6.1 Translation

Translation operation in the homogenous system can be represented as follow :

$$\begin{aligned} P' &= T \cdot P \\ \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \end{aligned}$$

$$x' = x + t_x$$

$$y' = y + t_y$$

$$1 = 1$$

With homogeneous representation, translation operation is converted to multiplicative operation.

4.6.2 Scaling

Scaling operation in a homogenous system can be represented as follow :

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

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

$$1 = 1$$

4.6.3 Rotation

Rotation operation in the homogenous system can be represented as follow :

$$\begin{aligned} P' &= R \cdot P \\ \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ x' &= x \cos \theta - y \sin \theta \\ y' &= x \sin \theta + y \cos \theta \\ 1 &= 1 \end{aligned}$$

4.7 Composite Transformation

- Composite transformation is any sequence of basic transformation operations.
- In practice, pure operations (translation, rotation or scaling) are rare. In interactive graphics packages, the designer often places the object on the screen and adjusts its size, location and orientation incrementally. The originally placed object goes through a sequence of various basic transformation operations.
- Combination of basic transformation operations is called composite transformation.
- We can generate a composite transformation matrix by multiplying matrices of basic transformation operations like translation, scaling, rotation, reflection, shearing etc.
- For column measure representation matrices are placed from right to left.
- Let's say M_1, M_2, \dots, M_n are the transformation matrices of some transformation operation in the same order, so for column measure representation of transformed coordinates is given as,

$$P' = M_n \dots M_2 \cdot M_1 \cdot P$$

Where M_i indicates i^{th} operation in sequence.

- For row measure representation, transformed coordinates are computed as follow :

$$P'^T = P^T \cdot M_1^T \cdot M_2^T \dots M_n^T$$

4.7.1 Successive Transformations

1. Translation

Let us translate the point (x, y) by $[t_{x1} \ t_{y1}]^T$ followed by $[t_{x2} \ t_{y2}]^T$. Here translation by $M_1 = [t_{x1} \ t_{y1}]^T$ is the first operation, which is followed by translation vector $M_2 = [t_{x2} \ t_{y2}]^T$.

So transformed coordinates are given as,

$$P' = M_2 \cdot M_1 \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Thus, two successive transformations are additive.

2. Rotation

Q. Prove that two successive rotations are additive i.e. $R(\theta_1) \cdot R(\theta_2) = R(\theta_1 + \theta_2)$

MU - May 18, 5 Marks

Q. Prove that two successive rotations are additive.

MU - May 19, 5 Marks

Let us rotate the point (x, y) by the angle θ_1 followed by angle θ_2 . Here rotation by angle θ_1 is the first operation, let's call the transformation matrix M_1 , which is followed by rotation with θ_2 angle, call the transformation matrix for this rotation M_2 . So transformed coordinates P' are given as,

$$P' = M_2 \cdot M_1 \cdot P = R(\theta_2) \cdot R(\theta_1) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2) & -\sin(\theta_1)\cos(\theta_2) - \sin(\theta_2)\cos(\theta_1) & 0 \\ \sin(\theta_1)\cos(\theta_2) + \sin(\theta_2)\cos(\theta_1) & \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = R(\theta_2 + \theta_1) \cdot P$$

\therefore Successive rotation is additive.

3. Scaling

By applying two successive scaling with scaling parameters $S_1 = [S_{x1} \ S_{y1}]^T$ and $S_2 = [S_{x2} \ S_{y2}]^T$ on point $[x \ y]^T$, we get

$$P' = S_2 \cdot S_1 \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_{x2}S_{x1} & 0 & 0 \\ 0 & S_{y2}S_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\therefore x' = x \cdot (S_{x2} S_{x1})$$

$$y' = y \cdot (S_{y2} S_{y1})$$

Thus, two successive scaling is multiplicative.

4.7.2 Examples

Ex. 4.7.1 : Write 3×3 transformation matrix for each of the following rotation about the origin (a) Counter clockwise rotation by 180° , (b) Clockwise rotation by 90° .

Soln. :

(a) 3×3 transformation matrix for counter-clockwise rotation by 180° about the origin.

$$M_1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos (180^\circ) & -\sin (180^\circ) & 0 \\ \sin (180^\circ) & \cos (180^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) 2×2 transformation matrix for clockwise rotation by 90° , So $\theta = -90^\circ$

$$M_2 = \begin{bmatrix} \cos (-90^\circ) & -\sin (-90^\circ) & 0 \\ \sin (-90^\circ) & \cos (-90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ex. 4.7.2 : Rotate a triangle defined by $A(0,0)$, $B(6,0)$ and $C(3,3)$ by 90° about the origin in the anticlockwise direction.

Soln. :

Here, rotation is to be performed with respect to origin by 90° in anticlockwise rotation. So $\theta = +90^\circ$.

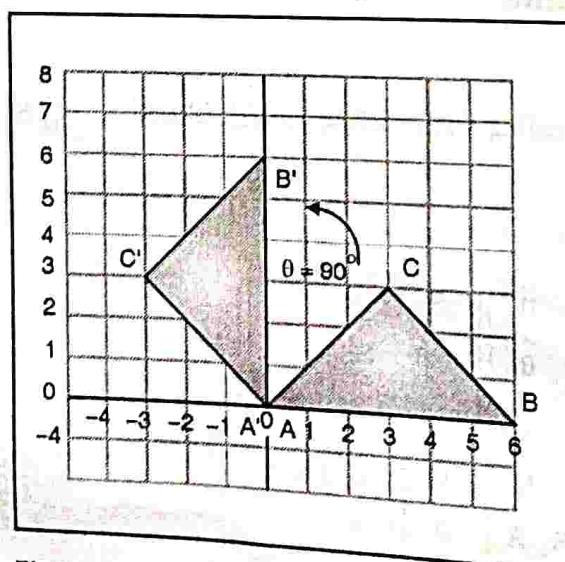


Fig. P. 4.7.2 : Output of said rotation operation

The transformed coordinates of the triangle are computed as,

$$P' = R \cdot P$$

$$\begin{aligned} &= \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) & 0 \\ \sin(90^\circ) & \cos(90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 6 & 3 \\ 0 & 0 & 3 \\ 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 6 & 3 \\ 0 & 0 & 3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -3 \\ 0 & 6 & 3 \\ 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Original Coordinates	Transformed Coordinates
A (0, 0)	A' (0, 0)
B (6, 1)	B' (0, 6)
C (3, 3)	C' (-3, 3)

Ex. 4.7.3 : Consider the square A(1,0), B(0,0), C(0,1), D(1,1). Rotate the square ABCD by 45° anticlockwise about point A(1,0).

Soln. :

Square is to be rotated about a reference point A(1, 0) by 45°. The desired operation is achieved by performing the following steps :

1. Translate reference point A(1, 0) to the origin.
2. Perform rotation by 45° in an anticlockwise direction.
3. Inverse translation of point A.

The transformation matrix for this sequence is given by $M = T^{-1} \cdot R_{(\theta = 45^\circ)} \cdot T$

$$P' = M \cdot P$$

$$\begin{aligned} P' &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45^\circ) & -\sin(45^\circ) & 0 \\ \sin(45^\circ) & \cos(45^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.709 & -0.709 & 0 \\ 0.709 & 0.709 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Multiply first two and last two matrices,

$$\begin{aligned} &= \begin{bmatrix} 0.709 & -0.709 & 1 \\ 0.709 & 0.709 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ P' &= \begin{bmatrix} 1 & 0.291 & -0.418 & 0.291 \\ 0 & -0.709 & 0 & 0.709 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Original Coordinates	Transformed Coordinates
A (1, 0)	A' (1, 0)
B (0, 0)	B' (0.291, - 0.709)
C (0, 1)	C' (- 0.418, 0)
D (1, 1)	D' (0.291, 0.709)

Ex. 4.7.4 : Perform 45° rotation of a triangle A(0, 0), B(1, 1) and C(5, 2). Find transformed coordinates after rotation, (a) About origin, (b) About P(-1, -1).

Soln. :

Triangle A(0, 0), B(1, 1), C(5, 2) is to be rotated by 45° .

(a) Rotation with respect to the origin.

$$\begin{aligned}
 P' &= M \cdot P = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{3}{\sqrt{2}} \\ 0 & \sqrt{2} & \frac{7}{\sqrt{2}} \\ 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

Original Coordinates	Transformed Coordinates
A (0, 0)	A' (0, 0)
B (1, 1)	B' (0, $\sqrt{2}$)
C (5, 2)	C' ($\frac{3}{\sqrt{2}}, \frac{7}{\sqrt{2}}$)

(b) Rotation with respect to P (-1, -1)

Rotated coordinates with respect to some reference point (x_r, y_r) is given by,

$$x' = x_r + (x - x_r) \cdot \cos \theta - (y - y_r) \cdot \sin \theta$$

$$y' = y_r + (x - x_r) \cdot \sin \theta + (y - y_r) \cdot \cos \theta$$

Let's consider $A'(x'_1, y'_1)$, $B'(x'_2, y'_2)$ and $C'(x'_3, y'_3)$

$$x'_1 = -1 + (0+1) \cdot \frac{1}{\sqrt{2}} - (0+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = -1$$

$$y'_1 = -1 + (0+1) \cdot \frac{1}{\sqrt{2}} + (0+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} = -1 + \sqrt{2}$$

$$x'_2 = -1 + (1+1) \cdot \frac{1}{\sqrt{2}} - (1+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{2}{\sqrt{2}} - \frac{2}{\sqrt{2}} = -1$$

$$y'_2 = -1 + (1+1) \cdot \frac{1}{\sqrt{2}} + (1+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{2}{\sqrt{2}} + \frac{2}{\sqrt{2}} = -1 + 2\sqrt{2}$$

$$x'_3 = -1 + (5+1) \cdot \frac{1}{\sqrt{2}} - (2+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{6}{\sqrt{2}} - \frac{3}{\sqrt{2}} = -1 + \frac{3}{\sqrt{2}}$$

$$y'_3 = -1 + (5+1) \cdot \frac{1}{\sqrt{2}} + (2+1) \cdot \frac{1}{\sqrt{2}} = -1 + \frac{6}{\sqrt{2}} + \frac{3}{\sqrt{2}} = -1 + \frac{9}{\sqrt{2}}$$

$$\therefore A' = (-1, -1 + \sqrt{2}), B' = (-1, -1 + 2\sqrt{2}), C' = \left(-1 + \frac{3}{\sqrt{2}}, -1 + \frac{9}{\sqrt{2}}\right)$$

Original Coordinates	Transformed Coordinates
A (0, 0)	A' (-1, -1 + \sqrt{2})
B (1, 1)	B' (-1, -1 + 2\sqrt{2})
C (5, 2)	C' \left(-1 + \frac{3}{\sqrt{2}}, -1 + \frac{9}{\sqrt{2}}\right)

Ex. 4.7.5 : Consider a triangle with vertices A(1,1), B(5,2) and C(3,4). Find out the transformation matrix which rotates given triangle about point C(3,4) by an angle 30° clockwise. Also, find the coordinates of the rotated triangle.

Soln. :

The given triangle is A(1, 1), B(5, 2), (3, 4). This triangle is to be rotated about C(3, 4) by 30° in a clockwise direction,

$$\text{So, } \theta = -30^\circ$$

Transformation sequence would be :

1. Translate C (3, 4) to the origin.
2. Rotate by 30° in a clockwise direction.
3. Inverse translation by shift-vector C.

$$\therefore \text{Transformation matrix, } M = T^{-1} \cdot R_{(\theta = -30^\circ)} \cdot T$$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-30^\circ) & -\sin(-30^\circ) & 0 \\ \sin(-30^\circ) & \cos(-30^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

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



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

$$= \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{2 - 3\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} & \frac{11 - 4\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Transformed coordinates, $P' = M \cdot P =$

$$\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{2 - 3\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} & \frac{11 - 4\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 5 & 3 \\ 1 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = M \cdot P = \begin{bmatrix} \frac{3 - 2\sqrt{3}}{2} & \frac{2\sqrt{3} + 4}{2} & 3 \\ \frac{10 - 3\sqrt{3}}{2} & \frac{6 - 2\sqrt{3}}{2} & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A (1, 1)	$A' \left(\frac{3 - 2\sqrt{3}}{2}, \frac{10 - 3\sqrt{3}}{2} \right)$
B (5, 2)	$B' \left(\frac{2\sqrt{3} + 4}{2}, \frac{6 - 2\sqrt{3}}{2} \right)$
C (3, 4)	C' (3, 4)

4.8 Reflection

- Almost all graphics packages provide functions for basic transformation operations like translation, scaling and rotation. Some packages also provide the facility of additional functions like reflection and shearing.
- **Reflection** is the transformation which produces a mirror image of an object about a given axis.
- Reflection is achieved by rotating an object by 180° around reference axis, perpendicular to the XY plane.
- Reflection does not alter the shape and size of the object, so this is a rigid body transformation.
- Let us discuss the various cases of reflection.

4.8.1 Reflection about X - Axis ($Y = 0$ Line)

- Reflection about X-axis is also known as a reflection about the $Y = 0$ line.
- Reflection about X-axis only flips y-coordinate, the x coordinate remains as it is. Point $P(2, -2)$ becomes $P'(2, 2)$ after reflection about X-axis.
- Reflection about X-axis is depicted in Fig. 4.8.1.

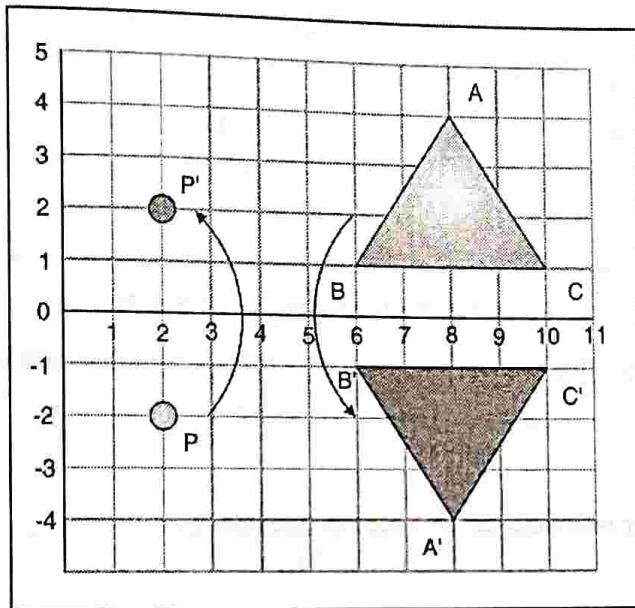


Fig. 4.8.1 : Reflection about the X-axis

- From Fig. 4.8.1, it is clear that,

$$\therefore x' = x$$

$$y' = -y$$

Transformation matrix for reflection about X axis is given as,

$$\text{Ref}_{(Y=0)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.8.2 Reflection about Y - Axis ($X = 0$ Line)

- Reflection about Y axis is also known as reflection about $X = 0$ line. Point $P(-2, -3)$ becomes $P'(2, -3)$ after reflection about Y-axis.
- Reflection about Y-axis is shown in Fig. 4.8.2.

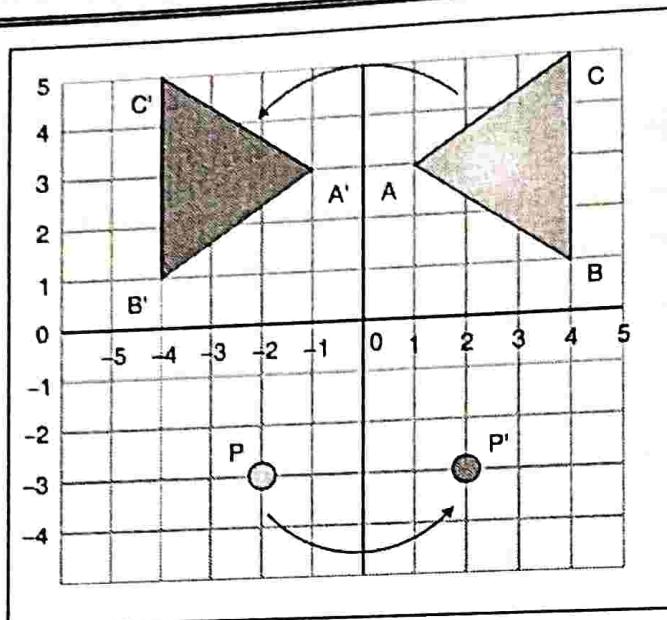


Fig. 4.8.2 : Reflection about the Y-axis

Reflection about Y-axis only flips x-coordinate, the y coordinate remains as it is.

$$\therefore x' = -x$$

$$y' = y$$

Transformation matrix for reflection about Y axis is given as,

$$\text{Ref}_{(x=0)} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.8.3 Reflection about X = Y Axis

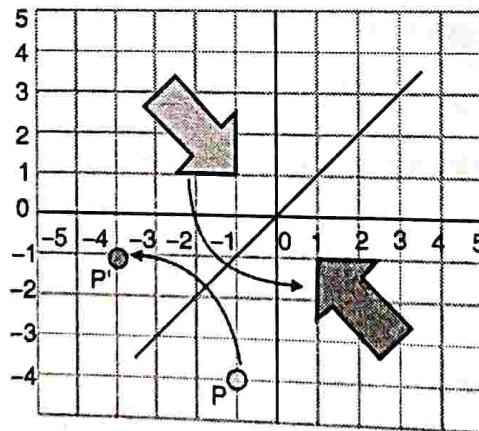


Fig. 4.8.3 : Reflection about X = Y axis

When a point (x, y) is reflected about a line $X = Y$, it becomes (y, x) . As shown in Fig. 4.8.3, $P(-1, -4)$ became $P'(-4, -1)$ after the reflection.

This operation swaps x and y coordinates.

$$\therefore x' = y$$

$$y' = x$$

Transformation matrix for reflection about $Y = X$ line is given as,

$$\text{Ref}_{(x=y)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.8.4 Reflection about $X = -Y$ Axis

- When point (x, y) is reflected about a line $X = -Y$, it becomes $(-y, -x)$. As shown in Fig. 4.8.4,

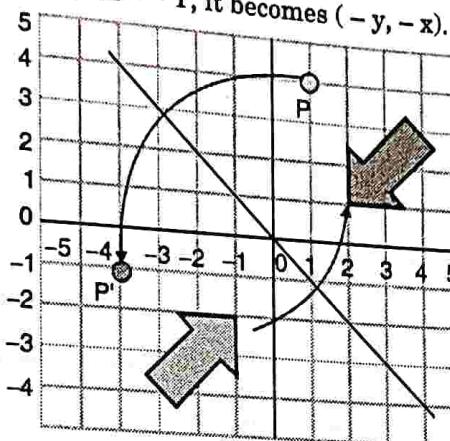


Fig. 4.8.4 : Reflection about $X = -Y$ axis

- $P(1, 4)$ became $P'(-4, -1)$ after the reflection.
- This operation will swap x and y coordinates and also change their sign.

$$\therefore x' = -y$$

$$y' = -x$$

- The transformation matrix for reflection about $Y = -X$ line is given as,

$$\text{Ref}_{(x=-y)} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Last both operations can be derived by aligning line $Y = X$ or $Y = -X$ with either of the principal axis and taking reflection about that particular axis followed by an inverse rotation.

4.8.5 Reflection about Origin

Rotation about the origin is carried out by flipping both, x and y coordinates.

This is 180° rotation about an axis that is perpendicular to the XY plane and that passes through the coordinate origin.

This operation flips both the coordinates.

$$\therefore x' = -x$$

$$y' = -y$$

The transformation matrix for reflection about the origin is given as,

$$\text{Ref}_{(\text{origin})} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.8.6 Reflection about any Line $y = mx + c$

Q. Explain the steps for 2D reflection w.r.t. line $y=mx$ and also derive a composite transformation Matrix.

MU - Dec. 19, 10 Marks

- Reflection about any arbitrary line $y = mx + c$ can be accomplished by the combination of translation, rotation and reflection.
- First, translate the line such that it passes through the origin.
- Then rotate the line so that it aligns with one of the principal axes
- Perform reflection operation
- Finally, restore the line to its actual position by performing inverse rotation and inverse translation.
- The composite transformation matrix for this sequence of operation is obtained by multiplying following matrices.

$$M = T^{-1} \cdot R^{-1} \cdot \text{Ref} \cdot R \cdot T$$

- We can implement reflections with respect to the coordinate axes or coordinate origin as scaling transformations with negative scaling factors.

4.8.7 Reflection about a Line Parallel to X-Axis

Reflection about a line parallel to the X-axis is carried out by following steps

Step 1 : Translate the line such that it aligns with X-axis (Refer Fig. 4.8.5(b)).

Step 2 : Perform the reflection about X-axis (Refer Fig. 4.8.5 (c)).

Step 3 : Translate back the line to its original position (Refer Fig. 4.8.5 (d)).

The sequence of operation is depicted in Fig. 4.8.5.

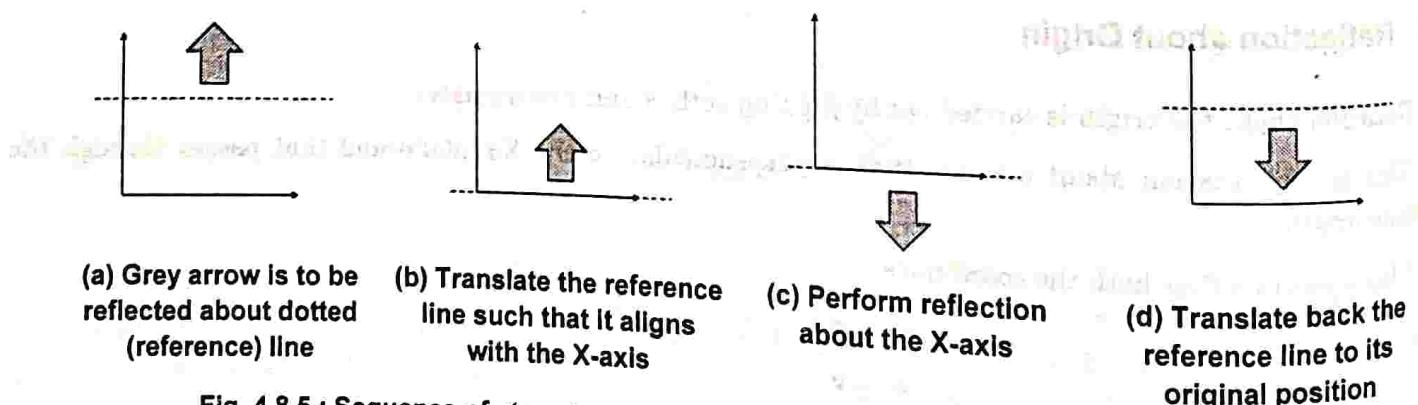


Fig. 4.8.5 : Sequence of steps to reflect object about a reference line parallel to X-axis

The transformation matrix for this operation is given by,

$$M = T^{-1} \cdot \text{Ref}_{(Y=0)} \cdot T$$

4.8.8 Reflection about a Line Parallel to Y-Axis

Reflection about a line parallel to the Y-axis is carried out by the following steps :

- Step 1 : Translate the line such that it aligns with Y-axis (Refer Fig. 4.8.6(b)).
- Step 2 : Perform the reflection about the Y-axis (Refer Fig. 4.8.6 (c)).
- Step 3 : Translate back the line to its original position (Refer Fig. 4.8.6 (d)).

The sequence of operation is depicted in Fig. 4.8.6.

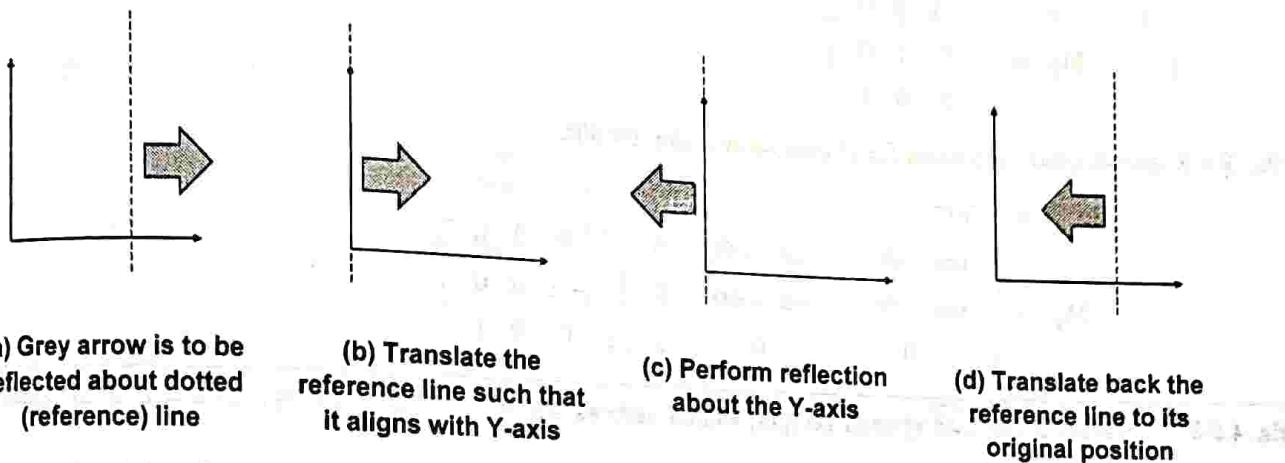


Fig. 4.8.6 : Sequence of steps to reflect object about a reference line parallel to the Y-axis

The transformation matrix for this operation is given by,

$$M = T^{-1} \cdot \text{Ref}_{(X=0)} \cdot T$$

Ex. 4.8.1 : Show that transformation matrix for a reflection about line $y = x$ is equivalent to reflection about X-axis followed by counter-clockwise rotation of 90° .

Soln. : Let's say M_1 is the transformation matrix for a reflection about line $y = x$

$$\therefore M_1 = \text{Ref}_{(y=x)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

And M_2 is the transformation matrix for reflection about X-axis followed by counter-clockwise rotation by 90° .

$$\begin{aligned} \therefore M_2 &= R(\theta = 90^\circ) \cdot \text{Ref}(y=0) \\ &= \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) \\ \sin(90^\circ) & \cos(90^\circ) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

Here, $M_1 = M_2$, hence given statement is proved.

Ex. 4.8.2 : Write 2×2 transformation matrix for each of the following rotation about the origin (a) Counter-clockwise rotation by 180° (b) Clockwise rotation by 90° .

Soln. :

(a) 2×2 transformation matrix for counter-clockwise rotation by 180° about the origin.

$$M_1 = \begin{bmatrix} \cos 0 & -\sin 0 & 0 \\ \sin 0 & \cos 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(180^\circ) & -\sin(180^\circ) & 0 \\ \sin(180^\circ) & \cos(180^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) 2×2 transformation matrix for clockwise rotation by 90° ,

$$\text{So, } \theta = -90^\circ$$

$$M_2 = \begin{bmatrix} \cos(-90^\circ) & -\sin(-90^\circ) & 0 \\ \sin(-90^\circ) & \cos(-90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ex. 4.8.3 : Reflect a diamond shaped polygon whose vertices are A(-1, 0), B(0, -2), C(1, 0) and D(0, 2) about the line $y = x + 2$.

Soln. :

Vertices of rectangle are A (-1, 0), B (0, -2), C (1, 0) and D(0, 2). Triangle is to be reflected about line $y = x + 2$.

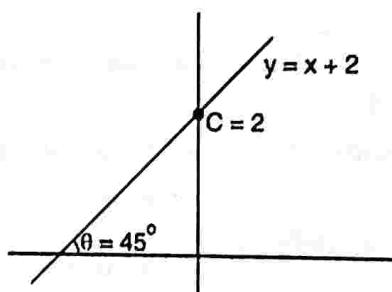


Fig. P.4.8.3

Compare $y = x + 2$ with $y = mx + c$

$$\therefore \tan \theta = m = 1$$

$$\theta = \tan^{-1}(1) = 45^\circ$$

$$\text{and } c = 2$$

- To perform the said operation, we should carry out the following steps.

1. Translate line by $(0, -2)$ so that it passes through the origin.
2. Rotate by 45° in a clockwise direction such that line gets align with the x-axis.

$$\theta = -45^\circ$$

3. Reflection about the x-axis.
 4. Inverse rotation.
 5. Inverse translation.
- \therefore Transformation matrix would be,

$$M = T^{-1} \cdot R_{(\theta)}^{-1} \cdot \text{Ref}_{(y=0)} \cdot R_{(\theta)} T$$

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Put, $\theta = -45^\circ$

$$\therefore M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

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

Transformed coordinates are given by,

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

Original Co-ordinates	Transformed Co-ordinates
A (-1, 0)	A' (-2, 1)
B (0, -2)	B' (-4, 2)
C (1, 0)	C' (-2, 3)
D(0, 2)	D' (0, 2)



4.9 Shearing

- A transformation that slants the shape of an object is called the shear transformation. Shearing is typically applied to soft objects like rubber, sheet, cloth etc.
- When we apply a horizontal force to the book which is kept upright, it bends in the direction of the force. Pages of the books will slide over each other. Top pages will be displaced more than the pages at the bottom.
- Shearing deforms the shape of an object and hence it is not rigid-body transformation.
- This deformation is proportional to shearing force and distance of the point of an object from the baseline.

4.9.1 X- Direction Shearing

- We can treat the object as a collection of layers. When we apply the horizontal force to the object, layers will slide. Neither the base layer alters nor the height of any internal layer changes.
- Effect of shearing in the X direction is depicted in Fig. 4.9.1.

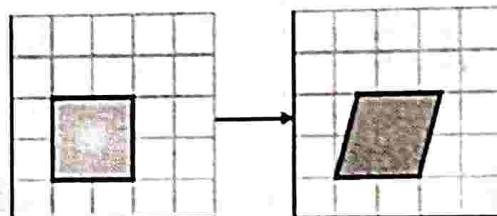


Fig. 4.9.1 : Shape deformation after shearing in the X direction

- It is intuitive that as the height of the internal layers in object increases from the base, its displacement would be more. So change in x-coordinate depends on two parameters :
 1. Height of internal layer from the base.
 2. The magnitude of the force.

So, transformed coordinates are given by

$$x' = x + Sh_x \cdot y$$

$$y' = y$$

- Where Sh_x is a shear parameter (force) in X-direction and y is the height of the internal layer from the base.
- The transformation matrix for X-direction shearing is,

$$SH_x = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation coordinates are computed as,

$$P' = SH_x \cdot P$$

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

Ex. 4.9.1: Apply shearing on cube with vertices A(0, 0), B(2, 0), C(2, 2) and D(0, 2), where $Sh_x = 3$.

Soln.:

For x-direction shearing with $y = 0$ as reference line

$$x' = x + Sh_x \cdot y$$

$$y' = y$$

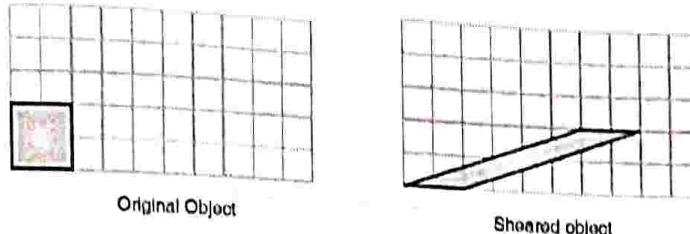


Fig. P. 4.9.1 : Shearing in the X direction

$$\text{For Point A, } x' = 0 + 3(0) = 0$$

$$\text{For Point B, } x' = 2 + 3(0) = 2$$

$$\text{For Point C, } x' = 2 + 3(2) = 8$$

$$\text{For Point D, } x' = 0 + 3(2) = 6$$

Original Co-ordinates	Transformed Co-ordinates
A (0, 0)	A' (0, 0)
B (2, 0)	B' (2, 0)
C (2, 2)	C' (8, 0)
D(0, 2)	D' (6, 0)

4.9.2 Shearing about a Line Parallel to X-axis

Like other operations, the previous transformation matrix is also position-dependent. The transformation matrix SH_x derived above is applicable only when the base of the object is on X-axis, means when the reference line is $Y = 0$.

We can derive generalized matrix for any reference line $Y = Y_{ref}$ parallel to X-axis using the following steps :

1. Translate line by $-Y_{ref}$ so the line gets aligned with X-axis.
2. Apply pure shear operation in the X-direction.
3. Translate line by $+Y_{ref}$ to bring it back to its original position.

∴ The composite transformation matrix would be, $M = T^{-1} \cdot SH_x \cdot T$

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & Y_{ref} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -Y_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & y_{ref} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & Sh_x & -y_{ref} \cdot Sh_x \\ 0 & 1 & -y_{ref} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & -y_{ref} \cdot Sh_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ex. 4.9.2 : Apply shearing on a unit square whose base is on line $y_{ref} = -2$ with $Sh_x = 2$.

Soln. :

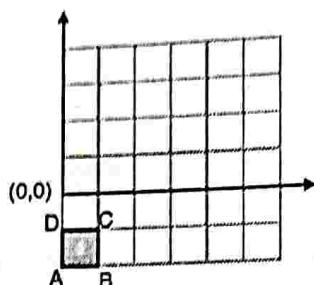


Fig. P. 4.9.2

$$P' = M \cdot P$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & -y_{ref} \cdot Sh_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\therefore x' = x + y \cdot Sh_x - y_{ref} \cdot Sh_x$$

$$y' = y$$

Y-coordinate of all point will remain as it is, the only x will change.

$$\text{For point A, } x' = 0 + 2(-2) - (-2)2 = -4$$

$$\text{For point B, } x' = 1 + 2(-2) - (-2)2 = 3$$

$$\text{For point C, } x' = 1 + (-1)2 - (-2)2 = 3$$

$$\text{For point D, } x' = 0 + (-1)2 - (-2)2 = 2$$

Position of a sheared cube is shown in Fig. P. 4.9.2(a).

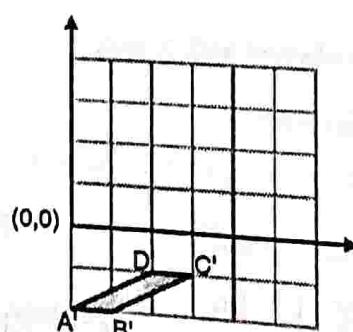


Fig. P. 4.9.2(a)

Original Coordinate	Transformed Coordinate
A(0, -2)	A' (0, -2)
B(1, -2)	B' (1, -2)
C(1, -1)	C' (3, -1)
D(0, -1)	D' (2, -1)

4.9.3 Y-Direction Shearing

- When shearing is applied in the y-direction, only y coordinates will change, x coordinates remain unchanged.
- The change in the y-direction is proportional to the magnitude of its x-coordinate and force applied in the y-direction.

$$\therefore x' = x$$

$$y' = y + x \cdot Sh_y$$

$$Sh_y = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fig. 4.9.2 depicts the result of shearing operation in Y-direction.

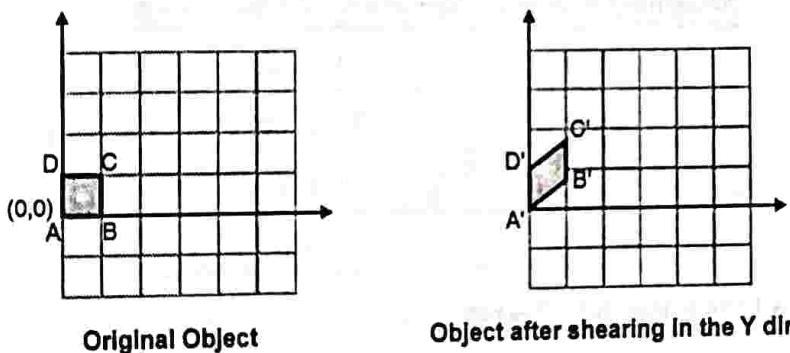


Fig. 4.9.2

Ex. 4.9.3 : Apply y-direction shearing on unit square with the base on $X = 0$ line and one corner at the origin with $Sh_y = 1/2$.

Soln. :

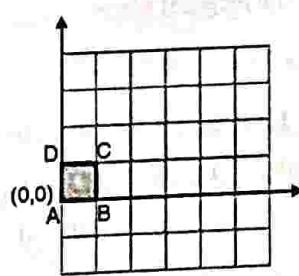


Fig. P. 4.9.3

Position of a unit cube is shown in the Fig. P. 4.9.3. Coordinates of vertices would be A(0, 0), B(1, 0), C(1, 1) and D(0, 1). For Y direction shear

$$\begin{aligned}x' &= x \\y' &= y + x \cdot Sh_y\end{aligned}$$

For point A, $y' = 0 + (0)1/2 = 0$

For point B, $y' = 0 + (1)1/2 = 1/2$

For point C, $y' = 1 + (1)1/2 = 3/2$

For point D, $y' = 1 + (0)1/2 = 1$

Position of sheared cube is shown in Fig. P. 4.9.3(a).

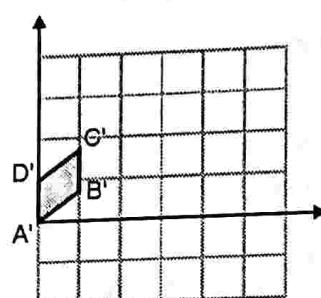


Fig. P. 4.9.3(a)

Original Coordinate	Transformed Coordinate
A(0, 0)	A' (0, 0)
B(0, 1)	B' (1, 1/2)
C(1, 1)	C' (1, 3/2)
D(1, 0)	D' (0, 1)

4.9.4 Shearing about a Line Parallel to Y-axis

We can derive generalized matrix for shearing about any reference line $X = X_{ref}$ parallel to Y-axis follows:

1. Translate line by $-X_{ref}$ so the line will align with Y-axis.
2. Apply pure shear operation in the Y direction.
3. Translate line by $+X_{ref}$ to bring it back to the original place.

∴ The composite transformation matrix $M = T^{-1} \cdot Sh_y \cdot T$

$$\begin{aligned}M &= \begin{bmatrix} 1 & 0 & x_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & x_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{ref} \\ Sh_y & 1 & -Sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & -Sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

Y-direction shear relative to line $X = x_{ref}$ can be produced with transformation matrix :

$$M = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & -Sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

- Ex. 4.9.4:** Convert the unit square to shifted parallelogram using x-direction shear transformation operation where parameter $Sh_x = \frac{1}{2}$ and $Y_{ref} = -1$ and unit square dimensions are (0, 0), (1, 0), (0, 1) and (1, 1).

Soln.:

Let us consider the vertices of unit square as A (0, 0), B (1, 0), C (1, 1) and D(0, 1). Shearing parameters are $Sh_x = \frac{1}{2}$ and $Y_{ref} = -1$.

Resultant co-ordinates are computed as,

$$P' = M \cdot P$$

$$M = \begin{bmatrix} 1 & Sh_x & -Y_{ref} \cdot Sh_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore P' = \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Original Co-ordinates	Transformed Co-ordinates
A (0, 0)	A' (-1/2, 0)
B (1, 0)	B' (1/2, 0)
C (1, 1)	C' (1, 1)
D (0, 1)	D' (0, 1)

- Ex. 4.9.5:** Apply the shearing transformation to Square with A(0,0), B(1,0), C(1,1) and D(0,1) as given below

(a) Shear parameter value of 0.5 relative to line $Y_{ref} = -1$

(b) Shear parameter value of 0.5 relative to line $X_{ref} = -1$

Soln.:

(a) Here, reference line is $Y_{ref} = 1$ and shear parameter is 0.5, so $Sh_x = 0.5$.

The transformation matrix for shearing with respect to reference line Y_{ref} is,

$$M_1 = \begin{bmatrix} 1 & Sh_x & -Y_{ref} \cdot Sh_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformed coordinates would be, $P' = M_1 \cdot P$

$$P' = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 1.5 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

(b) Here, reference line is $X_{ref} = -1$, and shear parameter is 0.5, so $Sh_y = 0.5$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & -X_{ref} \cdot Sh_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

\therefore Transformed coordinates are,

$$P' = M_2 P = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0.5 & 1 & 2 & 1.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Original coordinates	After shearing in the X direction	After shearing in the Y direction
A (0, 0)	A' (0.5, 0)	A' (0, 0.5)
B (1, 0)	B' (1.5, 0)	B' (1, 1)
C (1, 1)	C' (2, 1)	C' (1, 2)
D (0, 1)	D' (1, 1)	D' (0, 1.5)

Ex. 4.9.6 : Write matrices in a homogenous coordinate system for the 2D Shear transformations.

Soln. :

$$\text{The transformation matrix for shearing X direction, } SH_x = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{The transformation matrix for shearing about a line parallel to X-axis, } M = \begin{bmatrix} 1 & Sh_x & -y_{ref} \cdot Sh_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{The transformation matrix for shearing Y direction, } SH_y = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{The transformation matrix for shearing about a line parallel to Y-axis, } M = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & -Sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

4.10 Solved Examples

Ex. 4.10.1 : Translate a Square ABCD with the coordinates A(0, 0), B(5, 0), C(5, 5), D(0, 5) by 2 units in X-direction & 3 units in Y-direction.

Solt.:

Co-ordinates of square are A (0, 0), B (5, 0), C (5, 5) and D (0, 5). It is to be translated by 2 and 3 units in X and Y directions respectively.
 So, $t_x = 2$ and $t_y = 3$

Transformed coordinates are computed as,

$$P' = M \cdot P = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot P = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 5 & 5 & 0 \\ 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 2 & 7 & 7 & 2 \\ 3 & 3 & 8 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Original Co-ordinates	Transformed Co-ordinates
A (0, 0)	A' (2, 3)
B (5, 0)	B' (7, 3)
C (5, 5)	C' (7, 8)
D (0, 5)	D' (2, 8)

Ex. 4.10.2 : Find the sequence of transformation to scale the object with respect to point A in XY plane.

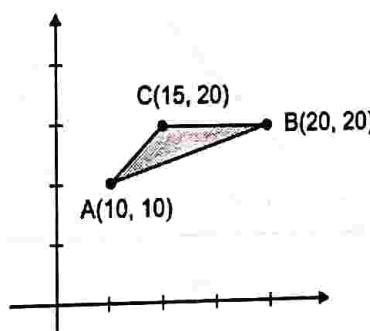


Fig. P. 4.10.2

Solt. :

Here, an object is to be scaled with respect to point A. so reference point is A (10, 10). Scaling with respect to the reference point is achieved by the following steps.

1. Translate reference point to the origin.
2. Scale the object
3. Inverse translation of reference point

\therefore Composite transformation $M = T^{-1} \cdot S \cdot T$

Consider S_x and S_y are the scaling parameters in X and Y direction respectively.



$$M = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

This is the required sequence of transformations.

Ex. 4.10.3 : Consider the triangle with vertices (10, 10), (40, 10), (30, 30). Apply scaling transformation with scale factor 5 in X and Y direction. Draw the triangle before and after transformation.

Soln. :

Let's consider the triangle A (10, 10), B (40, 10), C (30, 30).

Scaling factors in X and Y direction is 5.

$$\therefore s_x = s_y = 5$$

Scaling operation is carried out as,

$$P' = S \cdot P$$

$$P' = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 40 & 30 \\ 10 & 10 & 30 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 50 & 200 & 150 \\ 50 & 50 & 150 \\ 1 & 1 & 1 \end{bmatrix}$$

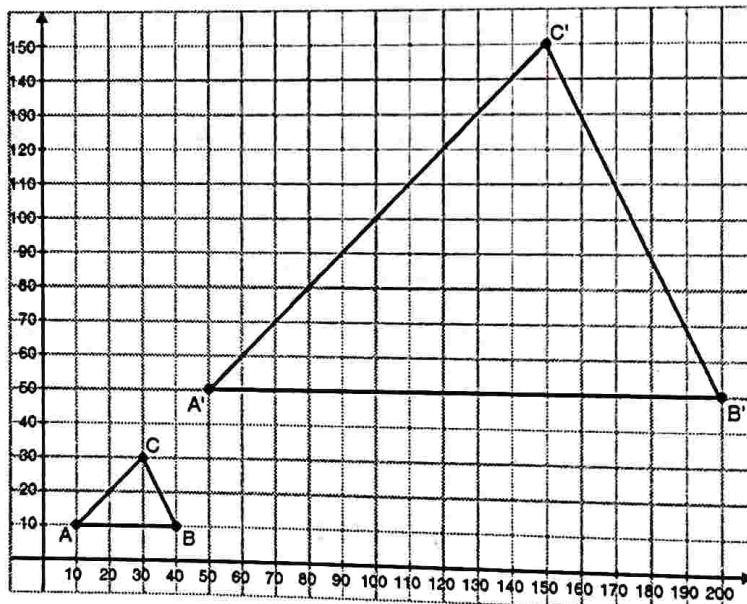


Fig. P. 4.10.3

Original Coordinates	Transformed Coordinates
A (10, 10)	A' (50, 50)
B (40, 10)	B' (200, 50)
C (30, 30)	C' (150, 150)

Ex. 4.10.4 : Develop a single transformation matrix which does the following on given object.

1. Reduce the size by $\frac{1}{2}$
2. Rotation about Y-axis by -30° .

Soln.: Here, the resultant matrix would be a composite transformation of scaling and rotation.

Given that, $S_x = S_y = S_z = \frac{1}{2}$ and $\theta = -30^\circ$

$$\therefore M = R_y(\theta = -30^\circ) \cdot S(S_x = S_y = S_z = \frac{1}{2})$$

$$\begin{aligned} M &= \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos(-30) & 0 & \sin(-30) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-30) & 0 & \cos(30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & -1/2 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{4} & 0 & -1/4 & 0 \\ 0 & 1/2 & 0 & 0 \\ 1/4 & 0 & \frac{\sqrt{3}}{4} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Ex. 4.10.5 : Rotate a triangle ABC by an angle 30° where the triangle has coordinates A(0, 0) B(10, 2) and C(7, 4).

Soln. :

Triangle is to be rotated about the origin by the angle $\theta = 30^\circ$. Let's assume P' is the set of rotated coordinates.

$$\begin{aligned} \therefore P' &= M \cdot P = R_{(\theta = 30)} \times P = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 10 & 7 \\ 0 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos(30^\circ) & -\sin(30^\circ) & 0 \\ \sin(30^\circ) & \cos(30^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 10 & 7 \\ 0 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\sqrt{3}}{2} & -1/2 & 0 \\ 1/2 & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 10 & 7 \\ 0 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & \frac{10\sqrt{3}-2}{2} & \frac{7\sqrt{3}-4}{2} \\ 0 & \frac{10+2\sqrt{3}}{2} & \frac{7+4\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Ex. 4.10.8 : Derive 2×2 transformation matrix for following operations :

1. Counter clockwise rotation by $\pi/2$.

2. Clockwise rotation by $-\pi$.

3. Reflection around $X = 0$ line followed by 180° counter clockwise direction.

4. Uniform scaling with $S_x = 2$ followed by reflection about $Y = -X$ line followed by 90° counter clockwise direction.

Soln. :

1. Counter-clockwise rotation by $\pi/2$:

Here, rotation direction is counter-clockwise, so θ is positive.

$$\theta = \pi/2$$

Transformation matrix,

$$M = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} \cos \pi/2 & -\sin \pi/2 \\ \sin \pi/2 & \cos \pi/2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2. Clockwise rotation by $-\pi$:

Here, rotation is in clockwise direction and rotation angle $\theta = -\pi$

Transformation matrix,

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

3. Reflection around $X = 0$ line followed by 180° counter clockwise direction :

Reflection about $X = 0$ line flips only X coordinate, it does not alter Y coordinate. Hence, the transformation matrix for reflection around $X = 0$,

$$M_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Transformation matrix for rotation by 180° in counterclockwise direction is

$$M_2 = \begin{bmatrix} \cos \pi & -\sin \pi \\ \sin \pi & \cos \pi \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

\therefore Resultant transformation matrix,

$$M = M_2 \cdot M_1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

4. Uniform scaling with $S_x = 2$ followed by reflection about $Y = -X$ line followed by 90° counter clockwise direction :

Composite transformation matrix for specified sequence of operation is derived as,

$$M = R_{(0 = 90^\circ)} \cdot \text{Ref}(Y = -X) \cdot S$$

$$M = \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) \\ \sin(90^\circ) & \cos(90^\circ) \end{bmatrix} \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

Ex. 4.10.7: Derive transformation matrix:

1. To increase the area of square 4 times.
2. To increase the size by four times.

Soln.: If we double the height and width of the square, its area becomes four times than original. So we need to perform scaling with $S_x = 2$ and $S_y = 2$.

$$\therefore \text{Resultant transformation matrix } M = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

To increase size four times, we have to multiply each side by 4.

$$\therefore S_x = S_y = 4$$

$$\therefore \text{Resultant transformation matrix, } M = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

Ex. 4.10.8: 1. Find transformation matrix to rotate the object about the origin by 45° in the counter-clockwise direction.

2. Find new coordinates of the point (8, 4) after rotation.

Soln.:

1. Here, rotation direction is counter-clockwise, so θ is positive.

$$\therefore \theta = 45^\circ$$

The transformation matrix is,

$$M = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} \cos(45^\circ) & -\sin(45^\circ) \\ \sin(45^\circ) & \cos(45^\circ) \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

2. Find new coordinates of the point (8, 4) after rotation.

Let us assume P and P' are original and transformed coordinates of the object respectively.

$$\therefore P' = M \cdot P = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 2\sqrt{2} \\ 6\sqrt{2} \end{bmatrix}$$

Ex. 4.10.9: A triangle is defined by $\begin{bmatrix} 2 & 4 & 4 \\ 2 & 2 & 4 \end{bmatrix}$. Find transformed coordinates after the following transformation

1. 90° rotation about the origin.
2. Reflection about line $X = Y$.

Soln.:

1. 90° rotation about origin :

$$\text{Here, } \theta = 90^\circ$$

Let us assume P and P' are original and transformed coordinates of the object respectively.



$$\therefore P' = M \cdot P = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 2 & 2 & 4 \end{bmatrix}$$

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

Original Coordinate	Transformed Coordinate
A(2, 2)	A'(-2, 2)
B(4, 2)	B'(-2, 4)
C(4, 4)	C'(-4, 4)

2. Reflection about line X = Y

Reflection about line X = Y

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

Original Coordinate	Transformed Coordinate
A(2, 2)	A'(2, -2)
B(4, 2)	B'(4, -2)
C(4, 4)	C'(4, -4)

Ex. 4.10.10 : Perform 90° rotation of triangle with vertices A(2, 2), B(4, 2) and C(3, 3)

1. About origin
2. About reference point (-2, 2)

Soln. :

1. Rotation about origin :

Here, rotation direction is counter-clockwise. So θ is positive.

$$\theta = 90^\circ$$

Let us assume P and P' are original and transformed coordinates of the object respectively.

$$\therefore P' = M \cdot P = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} 2 & 4 & 3 \\ 2 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 4 & 3 \\ 2 & 2 & 3 \end{bmatrix}$$

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

Original Coordinate	Transformed Coordinate
A(2, 2)	A'(-2, 2)
B(4, 2)	B'(-2, 4)
C(3, 3)	C'(-3, 3)

2. About reference point (-2, 2) :

The composite transformation matrix for rotation about some reference point is given as,

$$M = T^{-1} \cdot R \cdot T$$

Let us assume P and P' are original and transformed coordinates of the object respectively.

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

$$= \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 6 & 5 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -2 & -2 & -3 \\ 6 & 8 & 7 \\ 1 & 1 & 1 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A(2, 2)	A'(-2, 6)
B(4, 2)	B'(-2, 8)
C(3, 3)	C'(-3, 7)

Ex. 4.10.11 : Magnify a triangle with vertices A(0,0), B(1,1) and C(5,2) to twice of its size keeping B fixed.

Soln. :

This is fixed-point scaling with respect to reference point B(1, 1), so we need to perform following steps to carry out the desired operation.

1. Translate B to the origin.
2. Apply scaling with $S_x = S_y = 2$.
3. Inverse translation of B.

The composite transformation matrix for scaling with respect to some reference point is given as,

$$M = T^{-1} \cdot S \cdot T$$

Let us assume P and P' are original and transformed coordinates of the object respectively.

$$\therefore P' = M \cdot P$$

$$\therefore P' = T^{-1} \cdot S \cdot T \cdot P = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Multiplying first two and last two matrices

$$= \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 4 \\ -1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 9 \\ -1 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A(0, 0)	A'(-1, -1)
B(1, 1)	B'(1, 1)
C(5, 2)	C'(9, 3)

Ex. 4.10.12 : Comment on statement: "Two dimensional rotation and scaling are cumulative, if $S_x = S_y$ or $\theta = n\pi$ ".



Soln. :

$$\text{Matrix for 2D rotation, } M_1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Matrix for 2D scaling, } M_2 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We can say that operations are cumulative if $M_1M_2 = M_2M_1$.

(A) If $S_x = S_y$

Case 1 : Rotation followed by scaling

$$M_1M_2 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_x & 0 \\ 0 & 0 & 1 \end{bmatrix} (S_x = S_y) = \begin{bmatrix} S_x \cos \theta & -S_x \sin \theta & 0 \\ S_x \sin \theta & S_x \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Case 2 : Scaling followed by a rotation

$$M_2M_1 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x \cos \theta & -S_x \sin \theta & 0 \\ S_x \sin \theta & S_x \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here $M_1M_2 = M_2M_1$ So given statement is true for $S_x = S_y$.(B) If $\theta = n\pi$

Case 1 : Scaling followed by rotation

$$M_1M_2 = \begin{bmatrix} \cos(n\pi) & -\sin(n\pi) & 0 \\ \sin(n\pi) & \cos(n\pi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

 $(\sin(n\pi) = 0, \cos(n\pi) = 1)$

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

Case 2 : Rotation followed by scaling

$$M_2M_1 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(n\pi) & -\sin(n\pi) & 0 \\ \sin(n\pi) & \cos(n\pi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

Here, $M_1 M_2 = M_2 M_1$, So given statement is true for $\theta = n\pi$

Ex. 4.10.13 : The reflection along line $Y = X$ is equivalent to reflection along X-axis followed by counter-clockwise rotation by θ . Find the value of θ .

Soln. :

Transformation matrix for reflection about $Y = X$ line

$$M_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for reflection about X-axis

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for counter-clockwise rotation by angle θ

$$M_3 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As per given sequence of operation

$$M_1 = M_3 M_2$$

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

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

$$\therefore \cos \theta = 0, \sin \theta = 1$$

$$\therefore \theta = 90^\circ$$

Ex. 4.10.14 : Show that $\begin{bmatrix} \frac{1-t^2}{1+t^2} & \frac{2t}{1+t^2} \\ \frac{2t}{1+t^2} & \frac{1-t^2}{1+t^2} \end{bmatrix}$ represents pure rotation.

Soln. :

For pure rotation, the determinant of the rotation matrix is one.



$$\text{Given matrix is } M = \begin{bmatrix} \frac{1-t^2}{1+t^2} & \frac{2t}{1+t^2} \\ \frac{2t}{1+t^2} & \frac{1-t^2}{1+t^2} \end{bmatrix}$$

The determinant of given matrix M is :

$$|M| = \frac{1-t^2}{1+t^2} \cdot \frac{1-t^2}{1+t^2} - \frac{2t}{1+t^2} \cdot \frac{-2t}{1+t^2} = \frac{(1-t^2)^2 + 4t^2}{(1+t^2)^2} = \frac{(1+t^2)^2}{(1+t^2)^2} = 1$$

So the given matrix represents pure rotation.

Ex. 4.10.15: Show that two-dimensional reflection through X-axis followed by two-dimensional reflection through line $Y = -X$ is equivalent to a pure rotation about the origin by 270° .

Soln. :

Transformation matrix for two dimensional reflection through X-axis followed by two dimensional reflection through line $Y = -X$ is,

$$M_1 = \text{Ref}(Y = -X) \cdot \text{Ref}(Y = 0)$$

$$M_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix for rotation about origin is,

$$M_2 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here $\theta = 270^\circ$, so

$$M_2 = \begin{bmatrix} \cos(270^\circ) & -\sin(270^\circ) & 0 \\ \sin(270^\circ) & \cos(270^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here $M_1 = M_2$, so statement is proved.

Note: In general, two pure reflections about a line passing through origin is equivalent to the pure rotation.

Ex. 4.10.16: A mirror is placed vertically such that passes through points $(10, 0)$ and $(0, 10)$. Find the mirrored coordinate of a triangle with vertices A $(2, 2)$ B $(4, 2)$ and C $(3, 3)$.

Soln. :

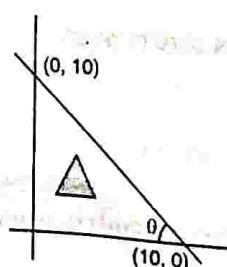


Fig. P. 4.10.16

Arrangement of object and line is shown in Fig. P. 4.10.16.

From Fig. P. 4.10.16,

$$\tan \theta = \frac{dy}{dx} = \frac{10}{10} = 1,$$

$$\text{so, } \theta = \tan^{-1}(1) = 45^\circ$$

To reflect the given triangle about given line, we have to align it with one of the four line, $x = 0, y = 0, x = y$ or $x = -y$.

Let's align it with X-axis. To align it with X-axis we should perform the following steps :

1. Translate (0, 10) to the origin.
2. Perform anticlockwise rotation by $\theta = 45^\circ$
3. Perform reflection about X-axis.
4. Inverse rotation
5. Inverse translation.

So the sequence of a resultant transformation matrix is,

$$M = T^{-1} \cdot R^{-1} \cdot \text{Ref}(Y=0) \cdot R \cdot T$$

$$\begin{aligned} M &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45^\circ & \sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45^\circ) & \sin(-45^\circ) & 0 \\ \sin(-45^\circ) & \cos(-45^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 10 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Transformed coordinates of the triangle are,

$$P' = M \cdot P = \begin{bmatrix} 0 & -1 & 10 \\ -1 & 0 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 3 \\ 2 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 8 & 7 \\ 8 & 6 & 7 \\ 1 & 1 & 1 \end{bmatrix}$$

Reflected coordinates of the triangle are A'(8, 8), B'(8, 6) and C'(7, 7)

Ex. 4.10.17 : Reflect triangle ABC with vertices A(2, 2), B(4, 2) and C(3, 3) around line $-3x + 4y - 8 = 0$.

Soln. :

From line equation $-3x + 4y - 8 = 0$

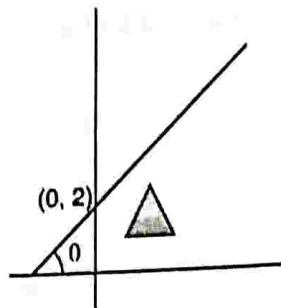


Fig. P. 4.10.17

$$4y = 3x + 8$$

$$\therefore y = \frac{3}{4}x + 2 = mx + c$$

$$\therefore m = \frac{3}{4}, c = 2$$

$$\tan \theta = \frac{3}{4}; \therefore \sin \theta = \frac{3}{5}; \cos \theta = \frac{4}{5}$$

Line makes a positive angle with X-axis (because $\tan \theta > 0$) and it intersects Y axis at point $(0, 2)$ ($c = 2$). Line orientation is shown in Fig. P. 4.10.17.

The transformation matrix for reflection is about the given line is achieved by the following steps :

1. Translate $(0, 2)$ to the origin.

2. Rotate line by $\theta = \tan^{-1}\left(\frac{3}{4}\right)$ angle in a clockwise direction.

3. Reflect triangle about X-axis.

4. Inverse rotation by θ .

5. Inverse translation.

$$\therefore M = T^{-1} \cdot R^{-1} \cdot \text{Ref}_{(Y=0)} \cdot R \cdot T$$

$$\begin{aligned}
 M &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4/5 & -3/5 & 0 \\ 3/5 & 4/5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4/5 & 3/5 & 0 \\ -3/5 & 4/5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 4/5 & -3/5 & 0 \\ 3/5 & 4/5 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4/5 & 3/5 & -6/5 \\ -3/5 & 4/5 & -8/5 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 4/5 & 3/5 & 0 \\ 3/5 & -4/5 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4/5 & 3/5 & -6/5 \\ -3/5 & 4/5 & -8/5 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7/25 & 24/25 & -48/25 \\ 24/25 & -7/25 & 64/25 \\ 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Coordinates of reflected triangle is given by,

$$P' = M \cdot P = \begin{bmatrix} 7/25 & 24/25 & -48/25 \\ 24/25 & -7/25 & 64/25 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 3 \\ 4 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} \frac{62}{25} & \frac{28}{25} & \frac{45}{25} \\ \frac{84}{25} & \frac{146}{25} & \frac{115}{25} \\ 1 & 1 & 1 \end{bmatrix}$$

$$\therefore A' = \left(\frac{62}{25}, \frac{84}{25} \right),$$

$$B' = \left(\frac{28}{25}, \frac{146}{25} \right),$$

$$C' = \left(\frac{45}{25}, \frac{115}{25} \right)$$

Original Coordinates	Transformed Coordinates
A(2, 4)	A' $\left(\frac{62}{25}, \frac{84}{25} \right)$
B(4, 2)	B' $\left(\frac{28}{25}, \frac{146}{25} \right)$
C(3, 3)	C' $\left(\frac{45}{25}, \frac{115}{25} \right)$

Review Questions

- Q.1 What is the transformation?
- Q.2 List out basic transformations techniques?
- Q.3 Explain 2D Transformation. Derive transformation matrix for the same.
- Q.4 Derive the suitable transformation matrix for translation.
- Q.5 Explain scaling transformation with respect to 2D.
- Q.6 What is scaling? Derive 2D transformation matrix for scaling with respect to origin and a reference point.
- Q.7 Derive the transformation matrix for scaling with respect to the origin.
- Q.8 Derive the transformation matrix for scaling with respect to the reference point.
- Q.9 Explain uniform and non-uniform scaling with suitable example.
- Q.10 Derive the transformation matrices for rotation with respect to origin and reference point.
- Q.11 Derive 2D transformation matrix for rotation with respect to the origin.
- Q.12 What do you mean by pure rotation matrix?
- Q.13 Derive 2D transformation matrix for rotation with respect to a reference point.
- Q.14 Explain in brief : Homogeneous coordinate.

 Computer Graphics (MU)

- Q. 15 What is the need of homogenous coordinate in transformation?
- Q. 16 List the properties of the homogeneous coordinate representation.
- Q. 17 Give homogenous coordinate matrix representation for translation, rotation and scaling.
- Q. 18 Give matrix representation for 2D translation.
- Q. 19 Give the homogeneous representation for translation, scaling and rotation.
- Q. 20 Give matrix representation for 2D scaling.
- Q. 21 Give matrix representation for 2D rotation.
- Q. 22 What is composite transformation?
- Q. 23 Find out the composite transformation matrix to rotate a given 2D object by an amount θ about the given point (x_r, y_r) .
- Q. 24 Derive transformation matrix for general pivot point rotation.
- Q. 25 Write a short note on : General pivot point rotation.
- Q. 26 Derive transformation matrix for general pivot point scaling.
- Q. 27 Write a short note on : General pivot point scaling.
- Q. 28 What is reflection? Derive transformation matrices for various cases of reflection.
- Q. 29 Derive transformation matrix for reflection about X axis.
- Q. 30 Write a short note on : Reflection about X axis.
- Q. 31 Derive transformation matrix for reflection about Y axis.
- Q. 32 Write a short note on : Reflection about Y axis.
- Q. 33 Derive transformation matrix for reflection about $X = Y$ axis.
- Q. 34 Write a short note on : Reflection about $X = Y$ axis.
- Q. 35 Derive transformation matrix for reflection about $X = -Y$ axis.
- Q. 36 Write a short note on : Reflection about $X = -Y$ axis.
- Q. 37 Derive transformation matrix for reflection about origin.
- Q. 38 Write a short note on Reflection about origin.
- Q. 39 Derive transformation matrix for reflection about any arbitrary line.
- Q. 40 How to perform reflection about a line $y = mx + c$.
- Q. 41 Derive transformation matrices for shearing in X and Y direction.
- Q. 42 How to perform shearing in X direction? Derive its transformation matrix.
- Q. 43 Derive transformation matrix for shearing with respect to line parallel to X-axis.
- Q. 44 How to perform shearing in Y direction? Derive Its transformation matrix.
- Q. 45 Derive transformation matrix for shearing with respect to line parallel to Y-axis.



Two-Dimensional Viewing and Clipping

Syllabus

Viewing transformation pipeline and Window to Viewport coordinate transformation

Clipping operations: Point clipping, Line clipping algorithms : Cohen-Sutherland, Liang : Barsky, Polygon Clipping Algorithms : Sutherland-Hodgeman, Weiler-Atherton.

5.1 2D Viewing

5.1.1 Viewing Transformation Pipeline

Q. What is window and view port?

MU - May 18, May 19, 5 Marks

- Graphics packages allow the user to specify which part of the scene is to be displayed.
- The process of selecting the part of the real-world scene to display it on some device is called **windowing**.
- **Clipping window** is the rectangular window against which visibility of scene object is tested.
- Clipping is the process of deciding and removing the portion of the object which is outside the clipping window.
- Part of the primitive outside the clipping window is not drawn. Clipping region may be of any arbitrary shape, but to simplify the clipping procedure, rectangle region is used.
- User can select single or clipping multiple windows simultaneously. Clipping reduces the computation by not processing the unnecessary part of the object.

There are two approaches for clipping the primitives :

1. **Scan conversion approach** : In this approach visibility of each pixel of an object is tested, and the pixel is drawn if it is visible. In other words, we perform the clipping during scan conversion of the object. This approach is simple but does extensive calculations.
 2. **Analytical approach** : In this approach, before scan converting the object, visible part of the primitives is calculated analytically. This approach does not perform the check for individual pixel. It reduces the computation but it is complex.
- The visible region of the scene inside the clipping window is displayed on some device. More formally, the world coordinate area which is selected for display is called a **window**.
 - An area on the display device to which window is mapped is called **viewport**.
 - Window defines *what* is to be displayed from the scene and viewport defines *where* it is to be displayed on the screen.
 - In most of the graphics package window and viewport is a rectangle, with edges parallel to the principal axis.



- The process of mapping the part of world coordinate scene to device coordinate is referred to as a viewing transformation or window to viewport transformation or windowing transformation.
- Fig. 5.1.1 illustrates the concept and relationship of window and viewport.

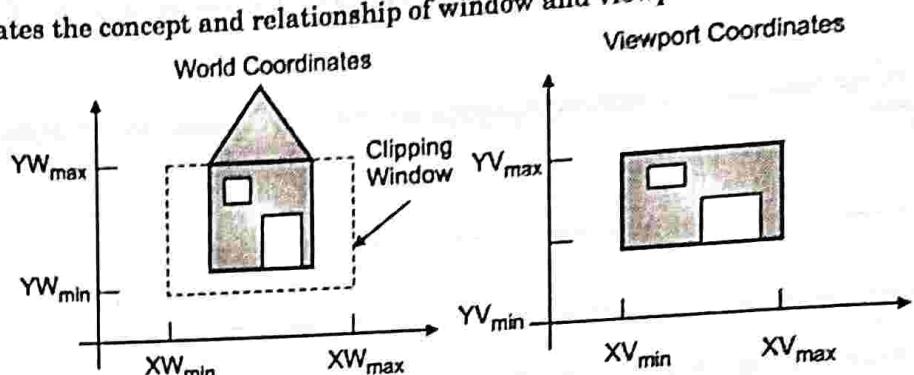


Fig. 5.1.1 : Window to view port transformation

- We can define the scene by creating objects with their actual dimensions, called **modelling coordinates**, or **local coordinates** or **master coordinates** i.e. (x_{mc}, y_{mc}) .
- Once the object is designed, it is placed at an appropriate position in the scene using **world coordinates** i.e. (x_{wc}, y_{wc}) .
- The scene may be displayed on different devices, so it must be mapped to the proper dimension to render it properly. World coordinates are then represented in **normalized coordinate** (x_{nc}, y_{nc}) , usually between 0 and 1. This makes the system independent of the various display devices.
- These normalized coordinate can be easily scaled to fit on any size of the display by multiplying it with a proper scaling factor that is known as **device coordinate** (x_{dc}, y_{dc}) . Fig. 5.1.2 shows sequence of coordinate transformations.

$$(x_{mc}, y_{mc}) \rightarrow (x_{wc}, y_{wc}) \rightarrow (x_{nc}, y_{nc}) \rightarrow (x_{dc}, y_{dc})$$

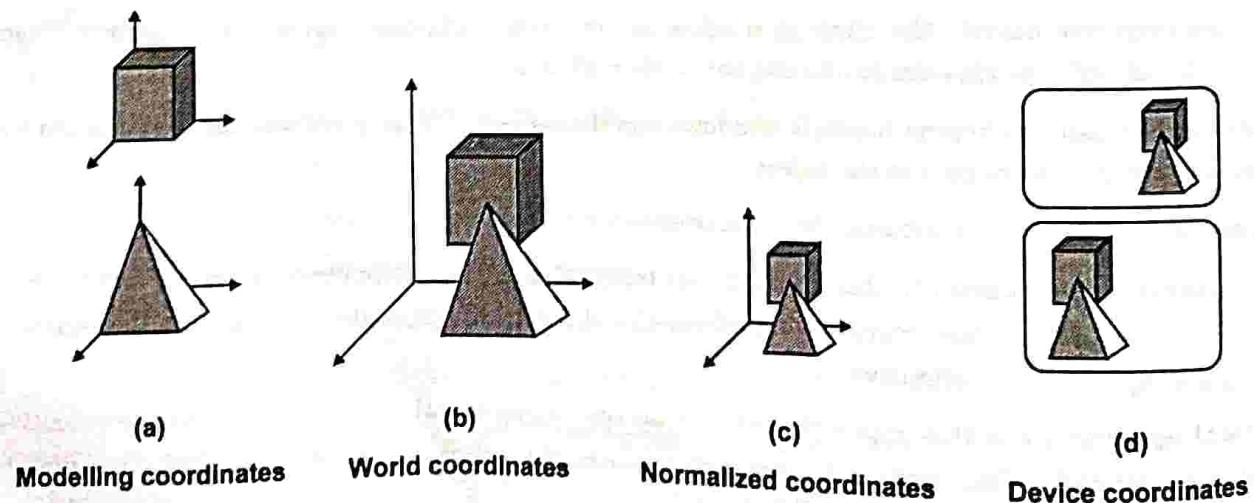


Fig. 5.1.2 : Coordinate transformation

- The viewing pipeline is depicted in Fig. 5.1.3.

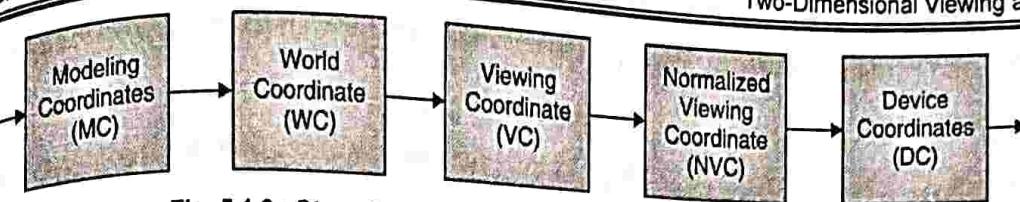


Fig. 5.1.3 : Steps for the window to viewport transformation

As shown in Fig. 5.1.4, we can view the scene on the device at different locations by changing the position of viewport.

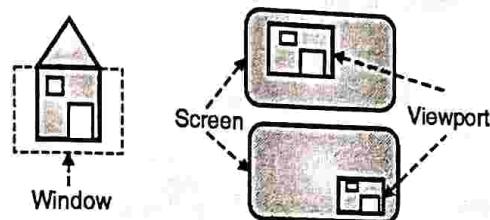


Fig. 5.1.4 : Mapping same window to different viewports

As shown in Fig. 5.1.4, the same window is mapped to viewports with different size and location on the screen.

- To achieve the zoom-in effect, we shall map window with different size on the fixed-size viewport. If the window size is made smaller, we get a zoom-in effect and if the window size is made larger, we get a zoom out effect.
- Panning is achieved by moving a fixed size window over various positions in the scene.

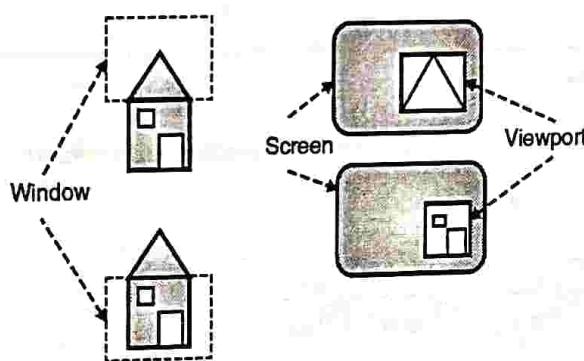


Fig. 5.1.5 : Effect of panning

5.1.2 Window to Viewport Coordinate Transformation

Q. Derive the window to view port transformation and also identify the geometric transformation involved.

MU - May 18, 5 Marks

Q. Derive the matrix for view port transformation.

MU - May 19, 5 Marks

- Physical description of the object goes through multiple sequences as shown in Fig. 5.1.6.
- Window to the viewport transformation is necessary because the size of the window and viewport may not be the same all the time. So actual scene selected by window needs to be rescaled to fit it in the viewport.



- Let (XW_{\min}, YW_{\min}) and (XW_{\max}, YW_{\max}) represent the lower left and upper-top corner points of clipping window, respectively.
- And let (XV_{\min}, YV_{\min}) and (XV_{\max}, YV_{\max}) represent the lower left and upper-top corner points of the viewport, respectively.
- As shown in Fig. 5.1.6, point (xw, yw) in the window is to be mapped to point (xv, yv) in the viewport.
- To maintain the same relative placement in the viewport as in a window, we normalize both.

$$\frac{xv - XV_{\min}}{XV_{\max} - XV_{\min}} = \frac{xw - XW_{\min}}{XW_{\max} - XW_{\min}}$$

$$xv - XV_{\min} = (XV_{\max} - XV_{\min}) \cdot \frac{xw - XW_{\min}}{XW_{\max} - XW_{\min}}$$

$$xv = XV_{\min} + (xw - XW_{\min}) \cdot S_x$$

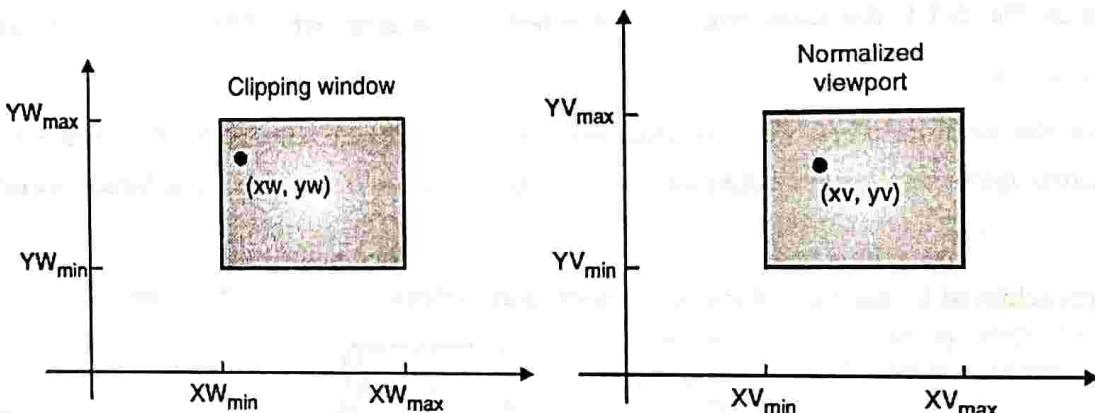


Fig. 5.1.6 : Relation between window and viewport

Similarly,

$$\frac{yv - YV_{\min}}{YV_{\max} - YV_{\min}} = \frac{yw - YW_{\min}}{YW_{\max} - YW_{\min}}$$

$$yv - YV_{\min} = (YV_{\max} - YV_{\min}) \cdot \frac{yw - YW_{\min}}{YW_{\max} - YW_{\min}}$$

$$yv = YV_{\min} + (yw - YW_{\min}) \cdot S_y$$

Where the scaling factors are,

$$S_x = \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}}$$

$$S_y = \frac{YV_{\max} - YV_{\min}}{YW_{\max} - YW_{\min}}$$

Matrix Representation

- If the lower-left corner of the window is not at the origin, we should first translate it to the origin before we map the points in the window to the viewport.
- Window to viewport transformation is achieved by the following three steps :
- (a) Translate the lower-left point of window to the origin.

$$\text{Translation matrix is defined as, } T = \begin{bmatrix} 1 & 0 & -XW_{\min} \\ 0 & 1 & -YW_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

- (b) Apply scaling to map the scene to the viewport.

$$\text{Scaling matrix is defined as, } S = \begin{bmatrix} \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}} & 0 & 0 \\ 0 & \frac{YV_{\max} - YV_{\min}}{YW_{\max} - YW_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- (c) Inverse translation in the viewport.

$$\text{Inverse translation matrix is defined as, } T^{-1} = \begin{bmatrix} 1 & 0 & XV_{\min} \\ 0 & 1 & YV_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

- The composite transformation matrix for the window to viewport transformation is given as,

$$M = T^{-1} \cdot S \cdot T$$

$$M = \begin{bmatrix} 1 & 0 & XV_{\min} \\ 0 & 1 & YV_{\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}} & 0 & 0 \\ 0 & \frac{YV_{\max} - YV_{\min}}{YW_{\max} - YW_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -XW_{\min} \\ 0 & 1 & -YW_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}} & 0 & \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}} + XV_{\min} \\ 0 & \frac{YV_{\max} - YV_{\min}}{YW_{\max} - YW_{\min}} & -XW_{\min} \cdot \frac{XV_{\max} - XV_{\min}}{XW_{\max} - XW_{\min}} + XV_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

5.2 2D Clipping

We can perform clipping operations to clip various primitives like point, line, polygon or even text. In the next few sections, we will discuss different algorithms to clip such basic primitives. Clipping operations determines the visible span of the primitive and maps it to the viewport.



The taxonomy for 2D clipping operations is described in Fig. 5.2.1.

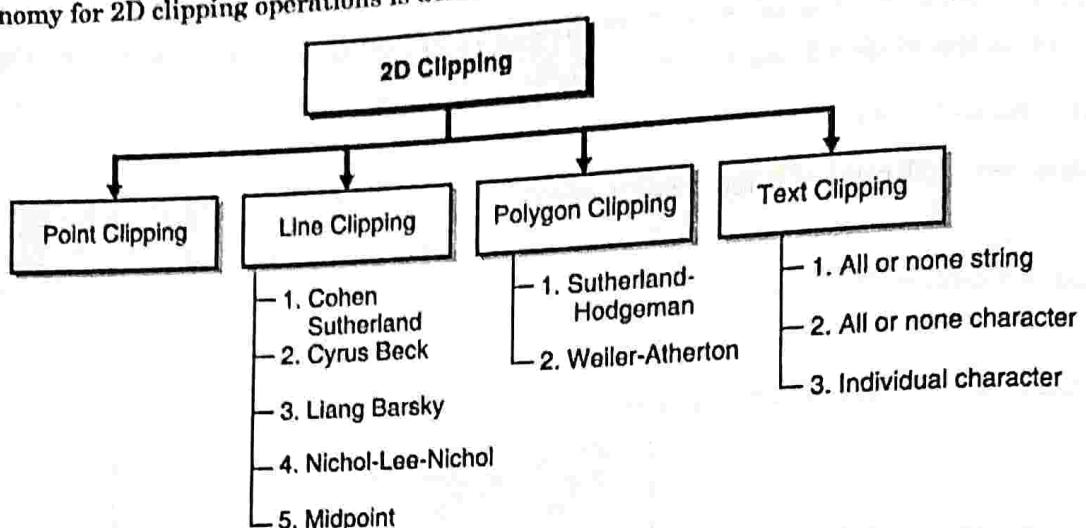


Fig. 5.2.1 : Taxonomy of 2D clipping operations

5.3 Point Clipping

- Before we discuss line clipping, let's look at the simpler problem of clipping individual point.

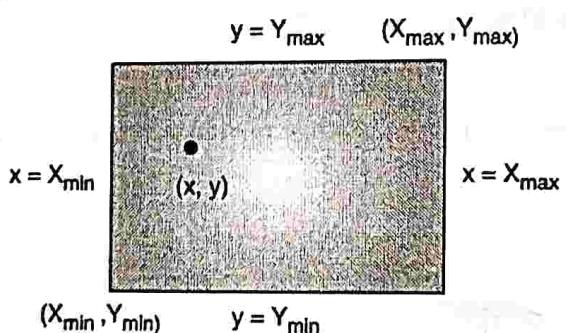


Fig. 5.3.1 : Point clipping

- Point clipping is very simple. It's a trivial case of acceptance or rejection. The point can be either fully inside or fully outside the clipping region. The point on the clipping window boundary is considered inside.
- Let, (X_{\min}, Y_{\min}) and (X_{\max}, Y_{\max}) represents the lower-left and top-right corners of the clipping window, respectively.
- Point (x, y) is inside the region only if the following four inequalities are true,

$$X_{\min} \leq x \leq X_{\max}$$

$$Y_{\min} \leq y \leq Y_{\max}$$

- For point being inside the clipping window, all four conditions must hold. If any of the four inequalities does not hold, the point is outside the clipping rectangle.

6.4 Line Clipping

- The line is widely used primitive in engineering drawing. Most of the engineering shapes contain lines.
- When we display the scene on the monitor screen, we should not render the portion of the scene which is outside the clipping window. It is essential to determine which portion of the line is inside the clipping region.
- Various cases of line membership are discussed here. Consider the clipping rectangle PQRS in Fig. 5.4.1.
- Scene contains four lines. If both endpoints of the line are within a rectangle, the line is completely visible. Otherwise, the line may be partially visible or completely outside the window.

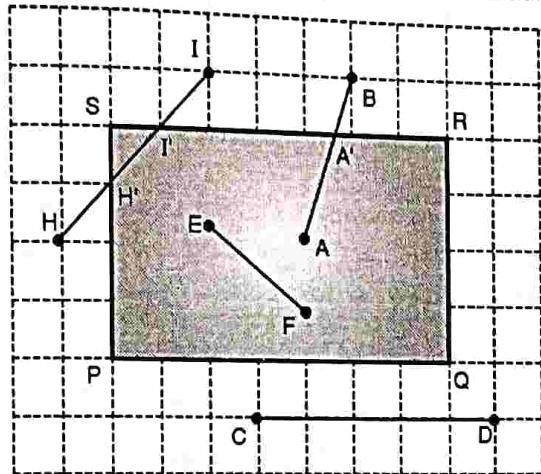


Fig. 5.4.1

- Line EF is completely visible and will be displayed as it is.
- One endpoint of line AB is outside the clipping region. The intersection point of line AB with the top edge is A'. Hence segment AA' will be displayed and A'B will be clipped.
- Line CD is completely outside the clipping window so it won't be displayed at all.
- Line HI intersects left and top border of clipping regions. Intersections with respective boundaries are H' and I'. So only segment H'I' will be displayed. Segments HH' and I'I would be clipped.
- Fig. 5.4.2 shows the output of the scene after the clipping procedure.

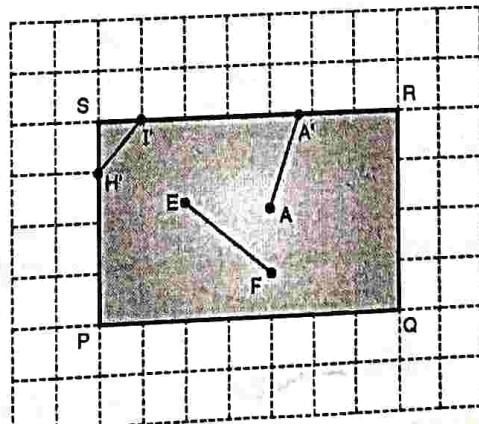


Fig. 5.4.2



5.4.1 Cohen - Sutherland Line Clipping Algorithm

Q. Explain the Cohen-Sutherland line clipping algorithm with suitable example.

MU - Dec. 18, 10 Marks

5.4.1.1 Working Mechanism

Cohen-Sutherland is a 2D line clipping algorithm. The main advantage of the algorithm is that it reduces the number of line intersections that must be calculated in scan conversion approach.

It operates in two phases :

1. Region code generation
2. Clipping

1. Region Code Generation

- It divides the plane into nine regions as shown in Fig. 5.4.4 and determines the visible portion of the line using outcodes (region code) of the endpoints. Outcode is a four-bit number. Values of these bits are set 1 if conditions shown in Fig. 5.4.3 are true for a particular bit.

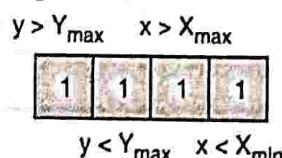


Fig. 5.4.3 : Setting the bits of outcode

- For any endpoint (x, y) of a line, the outcode is set according to the following conditions :

1001	1000	1010
0001	0000	0010
0101	0100	0110

Fig. 5.4.4 : Outcodes for all nine regions

Set first bit (most significant bit) if a point lies above window i.e. $y > Y_{\max}$.
Set second bit if a point lies below window i.e. $y < Y_{\min}$.

Set third bit if a point lies to the right of window i.e. $x > X_{\max}$.

Set fourth bit (least significant bit) if a point lies to left of window i.e. $x < X_{\min}$.

- If the point is inside the clipping window, none of the above conditions would be true. So, outcode of endpoint inside the clipping region would be 0000. Outcodes of all nine regions are shown in Fig. 5.4.4.

2. Clipping Procedure

- The algorithm quickly detects two trivial cases.
- If both endpoints of a line lie inside the window, the entire line is visible. It is trivially accepted and needs no clipping.

- On the other hand, if both endpoints of a line lie completely on one side of the window, the line lies completely outside the window. It is completely rejected.
- After calculating outcodes of both endpoints of the line, logical OR-AND conditions are evaluated to check trivial acceptance, rejection or partial visibility. We will discuss three possible cases.

Case I

- The line is completely inside clipping window if logical OR operation of outcodes yields to 0000. For example, line AB in Fig. 5.4.5 is trivially accepted.
- In short, if outcodes of both endpoints are 0000, the line is fully visible.

Case II

- If logical OR of the outcodes is not 0000, there are two possibilities, the line may be partially visible or it may be completely outside.

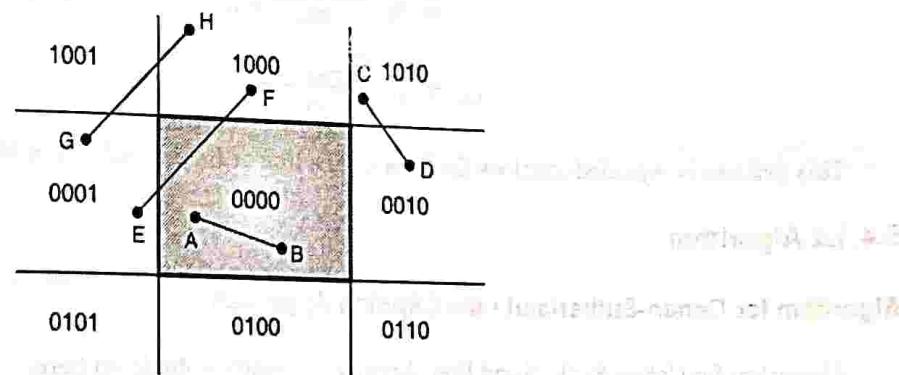


Fig. 5.4.5 : Different clipping cases

- The line is completely outside the clipping window if logical AND operation of outcomes does not yield to 0000.
- Above both tests are performed in the same order, i.e. Case II is applicable only if Case I is not true.
- For example, one endpoint of the line CD in Fig. 5.4.5 has outcode 1010 while the other endpoint has a code 0010. Logical OR is 1010, so it is not fully visible. The logical AND would be 0010 which indicates the line segment lies completely outside of the window.

Case III

- Outcodes of endpoints of the line, EF are 1000 and 0001. Logical OR of endpoints is 1001, so the line is not fully visible. The logical AND would be 0000, and the line could not be trivially rejected.
- In this case, we need to process line further. If logical AND operation of both endpoints is 0000, the line may or may not pass from clipping window. For example, like line EF, logical AND of outcodes of endpoints of line GH is also 0000, but that line is completely outside.
- If the line is not trivially accepted or rejected, then compare endpoints with window boundary to determine how much line segment can be discarded. The intersection point of line with clipping region edge can be computed as follow.

Consider the line with endpoints (x_1, y_1) and (x_2, y_2) . The slope of a line is given as,

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

- Using explicit representation of the line, $y = mx + c$, we can compute Y-Intercept of the line as, $c = y - mx$, where (x, y) is any point on the line. Both endpoints of the line are known, so we can put anyone in this equation to compute c , let's put (x_1, y_1) , so $c = y_1 - mx_1$.
- The Y-coordinate of an intersection with a vertical window boundary can be calculated as $y = mx + c$, where x is either X_{\min} or X_{\max} , depends on for which vertical line we are calculating Y.

$$Y = m \cdot X_{\min} + c \quad (\text{for intersection with left edge})$$

$$\text{Or } Y = m \cdot X_{\max} + c \quad (\text{for intersection with right edge})$$

Similarly, X-coordinate of an intersection with a horizontal window boundary is computed as,

$$X = (y - c) / m$$

Where y is either Y_{\min} or Y_{\max} , depends on for which horizontal line we are calculating X.

$$X = \frac{(Y_{\min} - c)}{m} \quad (\text{for intersection with the bottom edge})$$

$$\text{Or } X = \frac{(Y_{\max} - c)}{m} \quad (\text{for intersection with the top edge})$$

This process is repeated until we find the entire line segment within the window.

5.4.1.2 Algorithm

Algorithm for Cohen-Sutherland Line Clipping Approach

Algorithm for Cohen-Sutherland line clipping approach is depicted here.

Step 1 : Read (X_{\min}, Y_{\min}) and (X_{\max}, Y_{\max}) – Lower-left and top-right corner of clipping window

Step 2 : Read $A(x_1, y_1)$ and $B(x_2, y_2)$ end points of line

Step 3 : Compute outcode of A and B

if $y_1 > Y_{\max}$ then Outcode_A(1) = 1 else 0

if $y_2 > Y_{\max}$ then Outcode_B(1) = 1 else 0

if $y_1 < Y_{\min}$ then Outcode_A(2) = 1 else 0

if $y_2 < Y_{\min}$ then Outcode_B(2) = 1 else 0

if $x_1 > X_{\max}$ then Outcode_A(3) = 1 else 0

if $x_2 > X_{\max}$ then Outcode_B(3) = 1 else 0

if $x_1 < X_{\min}$ then Outcode_A(4) = 1 else 0

if $x_2 < X_{\min}$ then Outcode_B(4) = 1 else 0

Step 4 : if Outcode_A OR Outcode_B == 0000 then

Display entire line and goto step 5

else if Outcode_A AND Outcode_B ≠ 0000 then

Reject the entire line

else

Compute the intersection point with window boundaries

Repeat Step 4.

Step 5 : Stop

5.4.1.3 Pros and Cons

Advantages

- It is easy to understand.
- Simple to implement.
- Best suitable for the lines fully inside or outside.
- It can easily be extended for 3D line clipping.

Limitations

- Repeated clipping is expensive.
- Only applicable to rectangular clipping window. It cannot handle any other shape.
- It can be improved using more regions (e.g. Nichol Lee Nichol approach).

5.4.1.4 Examples

Ex 5.4.1: Use Cohen-Sutherland algorithm to clip two lines $P_1(40, 15) - P_2(75, 45)$ and $P_3(70, 20) - P_4(100, 10)$ against a window A(50,10), B (80,10), C (80,40) and D (50,40).

Soln.:

The spatial position of lines and rectangle is shown in Fig. P. 5.4.1.

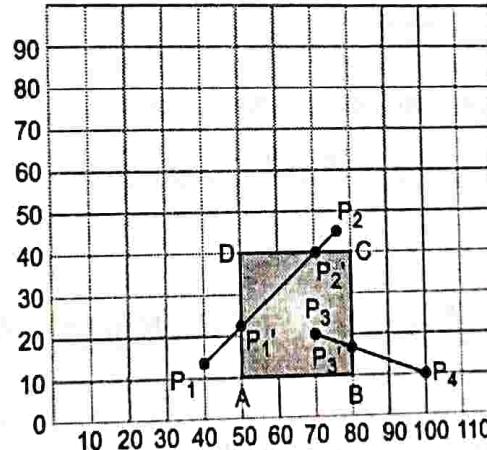


Fig. P. 5.4.1

Line P_1P_2

Given that, end points of line are $P_1(40, 15)$ and $P_2(75, 45)$

$$\text{Outcode}(P_1) = 0001 = A_1$$

$$\text{Outcode}(P_2) = 1000 = B_1$$

$$A_1 \text{ OR } B_1 = 0001 \text{ OR } 1000 = 1001$$

Logical OR of outcodes of both endpoints is not 0000, so the line is not completely visible.

$$A_1 \text{ AND } B_1 = 0001 \text{ AND } 1000 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

Outcode of endpoint A is 0001, that means line intersects left edge while moving from P_1 to P_2 . Let us call the intersection of point P'_1 . P'_1 is on line $x = x_{\min} = 50$. So its x-coordinate is 50, we have to compute only y-coordinate of P'_1 .

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{45 - 15}{75 - 40} = \frac{30}{35} = 0.857$$

$$\text{From, } y = mx + c, c = y - mx$$

By putting one of the points of line AB in this equation, we can find intercept of line with Y-axis. Let's put $P_1(40, 15)$ in above equation,

$$c = 15 - (0.857) \times (40) = 15 - 34.28 = -19.28$$

Put, $x = x_{\min} = 50$ in explicit line equation to compute corresponding Y-coordinate,

$$y = mx + c = 0.857 \times (50) - 19.28 = 42.85 - 19.28 = 23.57$$

$$\text{Thus, } P'_1 = (50, 23.57)$$

Outcode of endpoint P_2 is 1000, that means line intersects top edge. Let us call the intersection of point P'_2 . P'_2 is on line $y = 40$. So its y-coordinate is 40, we have to compute only x-coordinate of B' .

$$\text{From, } y = mx + c,$$

$$x = \frac{y - c}{m}$$

By putting the value of y of top edge in this equation, we can find the intersection x-coordinate of P'_2 .

$$x = \frac{40 + 19.28}{0.857} = \frac{59.28}{0.857} = 69.17$$

$$\text{Thus, } P'_2 = (69.17, 40)$$

So, line segment with endpoints $P'_1(50, 23.57)$ and $P'_2(69.17, 40)$ is the visible segment.

Line P_3P_4

Given that, endpoints of the line are $P_3(70, 20)$ and $P_4(100, 10)$

$$\text{Outcode}(P_3) = 0000 = C_1$$

$$\text{Outcode}(P_4) = 0010 = D_1$$

$$C_1 \text{ OR } D_1 = 0000 \text{ OR } 0010 = 0010$$

Logical OR of outcodes of both endpoint is not 0000, so the line is not completely visible.

$$C_1 \text{ AND } D_1 = 0000 \text{ AND } 0010 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10 - 20}{100 - 70} = \frac{-10}{30} = -0.33$$

From, $y = mx + c$, $c = y - mx$

By putting one of the points of line P_3P_4 in this equation, we can find intercept of line with Y-axis. Let's put $P_3(70, 20)$ in above equation,

$$c = 20 - (-0.33) \times (70) = 20 + 23.1 = 43.1$$

Outcode of endpoint P_4 is 0010, that means line intersects right edge. Let us call the intersection of point P'_3 .

From, $y = mx + c$,

By putting the value of x of a right edge in this equation, we can find the intersection y-coordinate of P'_3 .

$$\therefore y = (-0.33 \times 80) + 43.1 = -26.4 + 43.1 = 16.7$$

So, line segment with endpoints $P_3(70, 20)$ and $P'_3(80, 16.7)$ is the visible segment.

Original line Segments	Visible line Segments
$P_1(40, 15)$ and $P_2(75, 45)$	$P'_1(50, 23.57)$ and $P'_2(69.17, 40)$
$P_3(70, 20)$ and $P_4(100, 10)$	$P'_3(70, 20)$ and $P'_3(80, 16.7)$

Ex 5.4.2: Let R be the rectangular window whose lower left-hand corner at $L(-3, 1)$ and the upper right corner is at $R(2, 6)$. Find the region codes for the endpoints and use the Cohen-Sutherland algorithm to clip the line segments. Coordinates for line segments are,

For line AB, A(-4, 2) and B(-1, 7)

For line CD, C(-1, 5) and D(3, 8)

For line EF, E(-2, 3) and F(1, 2)

Soln.: The spatial position of lines and rectangle is shown in Fig. P. 5.4.2.

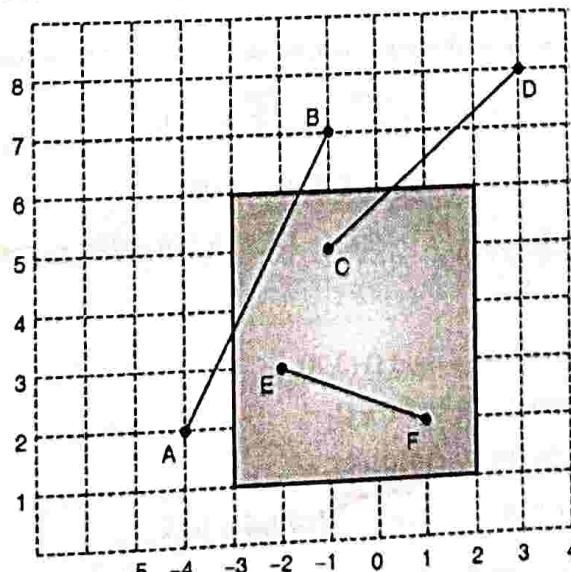


Fig. P. 5.4.2



Line AB

Given that, endpoints of the line are A (-4, 2) and B (-1, 7)

$$\text{Outcode (A)} = 0001 = A_1$$

$$\text{Outcode (B)} = 1000 = B_1$$

$$A_1 \text{ OR } B_1 = 0001 \text{ OR } 1000 = 1001$$

Logical OR of outcodes of both endpoints is not 0000, so the line is not completely visible.

$$A_1 \text{ AND } B_1 = 0001 \text{ AND } 1000 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

Outcode of endpoint A is 0001, that means line intersects left edge while moving from A to B. Let us call the intersection of point A with the left edge is A'. A' is on line $x = -3$. So its x coordinate is -3, we have to compute only y-coordinate of A'.

$$\text{Slope of line} = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7 - 2}{-1 + 4} = \frac{5}{3} = 1.67$$

$$\text{From, } y = mx + c, c = y - mx$$

By putting one of the points of line AB in this equation, we can find intercept of line with y axis. Let's put A (-4, 2) in above equation,

$$\therefore c = 2 - (1.67) \times (-4) = 2 + 6.68 = 8.68$$

$$\text{From, } y = mx + c = 1.67 \times (-3) + 8.68 = -5.01 + 8.68 = 3.67$$

Outcode of endpoint B is 1000, that means line intersects top edge while moving from A to B. Let us call the intersection of point B with the top edge is B'. B' is on line $y = 6$. So its y coordinate is 6, we have to compute only x-coordinate of B'.

$$\text{From, } y = mx + c,$$

$$x = \frac{y - c}{m}$$

By putting the value of y of top edge in this equation, we can find the intersection x-coordinate of B'.

$$\therefore x = \frac{6 - 8.68}{1.67} = \frac{-2.68}{1.67} = -1.60$$

$$\text{Outcode (A')} \text{ OR } \text{outcode (B')} = 0000 \text{ OR } 0000 = 0000$$

So line segment with endpoints A'(-3, 3.67) and B' (-1.6, 6) is the visible segment.

Line CD

Given that, end points of line are C (-1, 5) and D (3, 8)

$$\text{Outcode (C)} = 0000 = C_1$$

$$\text{Outcode (D)} = 1010 = D_1$$

$$C_1 \text{ OR } D_1 = 0000 \text{ OR } 1010 = 1010$$

Logical OR of outcodes of both end point is not 0000, so line is not completely visible.

$$C_1 \text{ AND } D_1 = 0000 \text{ AND } 1010 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.
Outcode of endpoint C is 0000, that means endpoint C is inside the clipping region.

$$\text{Slope of line } m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8 - 5}{3 + 1} = \frac{3}{4} = 0.75$$

$$\text{From, } y = mx + c, \quad c = y - mx$$

By putting one of the points of line CD in this equation, we can find intercept of line with the y-axis. Let's put C(-1, 5) in the above equation,

$$\therefore c = 5 - (0.75) \times (-1) = 5 + 0.75 = 5.75$$

Outcode of endpoint D is 1010, that means line intersects top and right edges while moving from A to B. Let us call the intersection of point D with the top edge is D'. D' is on line $y = 6$. So its y-coordinate is 6, we have to compute only x-coordinate of D'.

$$\text{From, } y = mx + c,$$

$$x = \frac{y - c}{m}$$

By putting the value of y of top edge in this equation, we can find the intersection x-coordinate of D'.

$$\therefore x = \frac{6 - 5.75}{0.75} = \frac{0.25}{0.75} = 0.33$$

$$\text{Outcode (C) OR outcode (D')} = 0000 \text{ OR } 0000 = 0000$$

So line segment with endpoints C(-1, 5) and D'(0.33, 6) is the visible segment.

Line EF

Given that, endpoints of the line are E(-2, 3) and F(1, 2)

$$\text{Outcode (E)} = 0000 = E_1$$

$$\text{Outcode (F)} = 0000 = F_1$$

$$E_1 \text{ OR } F_1 = 0000 \text{ OR } 0000 = 0000$$

Logical OR of outcodes of both endpoints is 0000, so the line is completely visible. It does not require any clipping.

Original line Segments	Visible line segments
A(-4, 2) and B(-1, 7)	A'(-3, 3.67) and B'(-1.6, 6)
C(-1, 5) and D(3, 8)	C(-1, 5) and D'(0.33, 6)
E(-2, 3) and F(1, 2)	E(-2, 3) and F(1, 2)

Ex. 6.4.3 : Let ABCD be the rectangular window with A(20, 20), B(90, 20), C(90, 70) and D(20, 70). Find region codes for endpoints and use the Cohen-Sutherland algorithm to clip the lines:

- (i) P₁P₂ with P₁(10, 30), P₂(80, 90)
- (ii) Q₁Q₂ with Q₁(10, 10), Q₂(70, 60)



Soln. :

The spatial position of lines and rectangle is shown in Fig. P. 5.4.3.

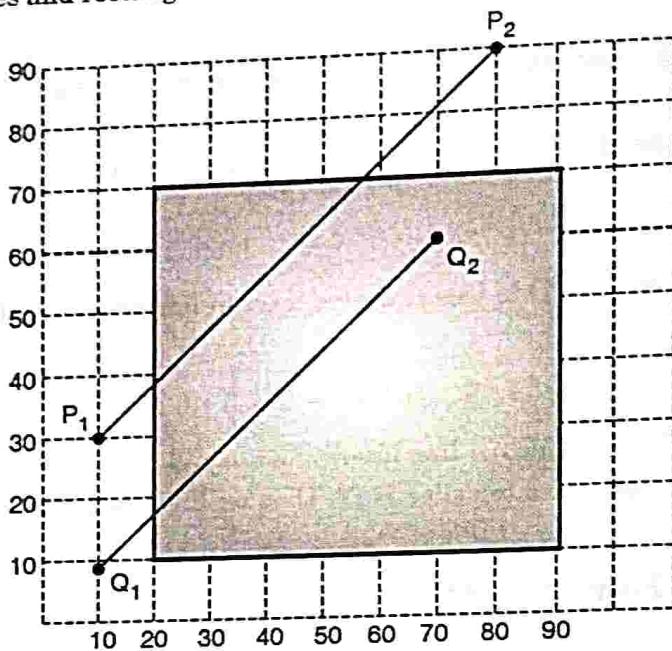


Fig. P. 5.4.3

Line P_1P_2

Given that, end points of line are $P_1(10, 30)$ and $P_2(80, 90)$

$$\text{Outcode}(P_1) = 0001 = X$$

$$\text{Outcode}(P_2) = 1000 = Y$$

$$X \text{ OR } Y = 0001 \text{ OR } 1000 = 1001$$

Logical OR of outcodes of both endpoints is not 0000, so the line is not completely visible.

$$X \text{ AND } Y = 0001 \text{ AND } 1000 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

Outcode of endpoint, P_1 is 0001, that means line intersects left edge while moving from P_1 to P_2 . Let us call the intersection of point P_1 with the left edge is P'_1 . P'_1 is on line $x = 20$. So its x-coordinate is 20, we have to compute the only y-coordinate of P'_1 .

$$\text{Slope of line} = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90 - 30}{80 - 10} = \frac{60}{70} = 0.86$$

$$\text{From, } y = mx + c, \quad c = y - mx$$

By putting one of the points of line P_1P_2 in this equation, we can find intercept of line with the y-axis
Let's put $P_1(10, 30)$ in above equation,

$$\therefore c = 30 - (0.86) \times (10) = 30 - 8.6 = 21.4$$

$$\text{From, } y = mx + c = 0.86 \times (20) + 21.4 = 17.2 + 21.4 = 38.6$$

Outcode of end point P_2 is 1000, that means line intersects top edge while moving from P_1 to P_2 . Let us call the intersection of point P_2 with top edge is P'_2 . P'_2 is on line $y = 70$. So its y-coordinate is 70, we have to compute only x-coordinate of P'_2 .

$$\text{From, } y = mx + c,$$

$$x = \frac{y - c}{m}$$

By putting the value of y of top edge in this equation, we can find the intersection x-coordinate of P'_2 .

$$\therefore x = \frac{70 - 21.4}{0.86} = \frac{48.6}{0.86} = 56.51$$

$$\text{Outcode (A') OR outcode (B')} = 0000 \text{ OR } 0000 = 0000$$

So line segment with endpoints $P'_1 (20, 38.6)$ and $P'_2 (56.51, 70)$ is the visible segment.

Line $Q_1 Q_2$

Given that, endpoints of the line are $Q_1 (10, 10)$ and $Q_2 (70, 60)$

$$\text{Outcode (Q}_1\text{)} = 0101 = X$$

$$\text{Outcode (Q}_2\text{)} = 0000 = Y$$

$$X \text{ OR } Y = 0101 \text{ OR } 0000 = 0101$$

Logical OR of outcodes of both endpoint is not 0000, so the line is not completely visible.

$$X \text{ AND } Y = 0101 \text{ AND } 0000 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

Outcode of endpoint Q_1 is 0101, that means line intersects bottom and left edges while moving from Q_1 to Q_2 . Let us call the intersection of point Q_1 with the bottom edge is Q'_1 . Q'_1 is on line $y = 20$. So its y-coordinate is 20, we have to compute only x-coordinate of Q'_1 .

$$\text{Slope of line} = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{60 - 10}{70 - 10} = \frac{50}{60} = 0.83$$

$$\text{From, } y = mx + c, \quad c = y - mx$$

By putting one of the points of line $P_1 P_2$ in this equation, we can find intercept of line with the y-axis. Let's put $Q_1 (10, 10)$ in above equation,

$$\therefore c = 10 - (0.83) \times (10) = 10 - 8.3 = 1.7$$

$$\text{From, } y = mx + c,$$

$$x = \frac{y - c}{m}$$

By putting value of y of top edge in this equation, we can find the intersection x-coordinate of Q'_1 .

$$\therefore x = \frac{20 - 1.7}{0.83} = \frac{18.3}{0.83} = 22.04$$

$$\text{Outcode (Q}'_1\text{)} \text{ OR outcode (Q}_2\text{)} = 0000 \text{ OR } 0000$$

$$= 0000$$



So line segment with endpoints $Q'_1(22.04, 20)$ and $P'_2(70, 60)$ is the visible segment.

Original line Segments	Visible line segments
$P_1(10, 30)$ and $P_2(80, 90)$	$P'_1(20, 38.6)$ and $P'_2(56.51, 70)$
$Q_1(10, 10)$ and $Q_2(70, 60)$	$Q'_1(22.04, 20)$ and $Q'_2(70, 60)$

Ex. 5.4.4 : Clip the line PQ having coordinates $P(4, 1)$ and $Q(6, 4)$ against the clip window having vertices A(3, 2), B(7, 2), C(7, 6) and D(3, 6) using Cohen Sutherland line clipping algorithm. MU - Dec. 18, 10 Marks

Soln. :

Given that, endpoints of the line are P (4, 1) and Q (6, 4)

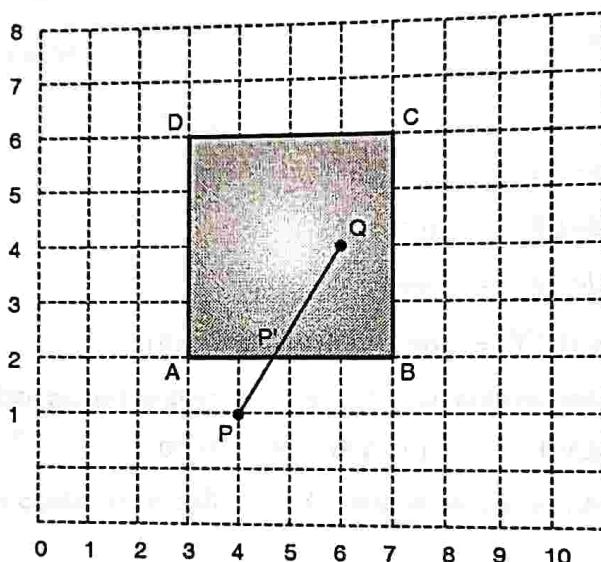


Fig. P. 5.4.4

The line is to be clipped against rectangle A (3, 2), B (7, 2), C (7, 6) and D (3, 6).

The spatial position of the line and rectangle is shown in Fig. P. 5.4.4.

$$\text{Outcode}(P) = 0100 = P_1$$

$$\text{Outcode}(Q) = 0000 = Q_1$$

$$P_1 \text{ OR } Q_1 = 0100 \text{ OR } 0000 = 0100$$

Logical OR of outcodes of both endpoints is not 0000, so the line is not completely visible.

$$P_1 \text{ AND } Q_1 = 0100 \text{ AND } 0000 = 0000$$

Logical AND of outcodes of both endpoint is 0000. So the line is not completely outside. Let us decide the visible portion of the line.

Outcode of endpoint P is 0100, that means line intersects bottom edge while moving from P to Q. Let us call the intersection point of P with the bottom edge is P'. P' is on line $y = 7$. So its y-coordinate is 7, we have to compute only x-coordinate of P'.

$$\text{Slope of line} = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4 - 1}{6 - 4} = \frac{3}{2} = 1.5$$

$$\text{From, } y = mx + c, \quad c = y - mx$$

By putting one of the points of line PQ in this equation, we can find intercept of line with y axis. Let's put $P(4, 1)$ in above equation,

$$\therefore c = 1 - (1.5) \times (4) = 1 - 6 = -5$$

$$\text{From, } y = mx + c$$

$$x = \frac{y - c}{m}$$

By putting the value of y of a bottom edge in this equation, we can find the intersection x-coordinate of P' .

$$\therefore x = \frac{2 - (-5)}{1.5} = \frac{7}{1.5} = 4.67$$

\therefore Coordinate of P' would be (4.67, 2).

outcode (P') OR outcode (Q) = 0000 OR 0000 = 0000

So line segment $P'Q$ is the visible segment.

Original line Segments	Visible line segments
$P(4, 1)$ and $Q(6, 4)$	$P'(4.67, 2)$ and $Q(6, 4)$

5.4.2 Liang - Barsky Line Clipping Algorithm

Q. Explain Liang Barsky line clipping algorithm, what is its benefit over CohenSutherland algorithm?

MU - Dec. 19, 5 Marks

5.4.2.1 Working Mechanism

- The Liang-Barsky algorithm uses the parametric equation of a line. It is also known as parametric line clipping approach.
- Cohen-Sutherland is good at trivial acceptance and rejection cases. Liang Barsky algorithm is significantly more efficient than Cohen-Sutherland when actually clipping is required.
- Let's consider the line with endpoints $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$. Parametric representation of line is given as,

$$x = x_0 + (x_1 - x_0) \cdot u = x_0 + u \cdot \Delta x$$

$$y = y_0 + (y_1 - y_0) \cdot u = y_0 + u \cdot \Delta y$$

- Where, u is a parameter that controls the line points, $0 \leq u \leq 1$
- Point is inside the clipping region only if it holds following inequalities :

$$X_{\min} \leq x_0 + u \cdot \Delta x \leq X_{\max} \quad \dots(5.4.1)$$

$$Y_{\min} \leq y_0 + u \cdot \Delta y \leq Y_{\max} \quad \dots(5.4.2)$$

By re writing above inequalities,

From Equation (5.4.1),

$$X_{\min} - x_0 \leq u \cdot \Delta x \leq X_{\max} - x_0$$

First inequality in above equation is, $X_{\min} - x_0 \leq u \cdot \Delta x$,

$$-u \cdot \Delta x \leq x_0 - X_{\min} \quad \dots(5.4.3)$$

Second inequality is, $u \cdot \Delta x \leq X_{\max} - x_0 \quad \dots(5.4.4)$

Similarly, from Equation (5.4.2),

$$Y_{\min} - y_0 \leq u \cdot \Delta y \leq Y_{\max} - y_0$$

First inequality in above equation is,

$$Y_{\min} - y_0 \leq u \cdot \Delta y, \quad \dots(5.4.5)$$

$$-u \cdot \Delta y \leq Y_0 - Y_{\min}$$

Second inequality is,

$$u \cdot \Delta y \leq Y_{\max} - y_0 \quad \dots(5.4.6)$$

Equation (5.4.3) to (5.4.6) can be written as,

$$P_k \cdot u \leq q_k, k = 1, 2, 3, 4$$

Where,

$p_1 = -\Delta x$	$q_1 = x_0 - X_{\min}$	Left
$p_2 = \Delta x$	$q_2 = X_{\max} - x_0$	Right
$p_3 = -\Delta y$	$q_3 = y_0 - Y_{\min}$	Bottom
$p_4 = \Delta y$	$q_4 = Y_{\max} - y_0$	Top

To compute the final visible segment, we do the following calculations :

- If $p_k = 0$, line is parallel to the boundary of the clipping region. With $p_k = 0$, if $q_k < 0$, the line is completely outside the clipping region and line can be completely clipped. When $p_k < 0$ the line proceeds outside to inside the clip window for a k^{th} boundary. Similarly, when $p_k > 0$, the line proceeds inside to outside for the k^{th} boundary. Value of $k = 1, 2, 3, 4$ corresponds to left, right, bottom and top edge.
- The line can enter or leave the region maximum two times. So for entering intersections, we should select the highest value of parameter u , and for leaving intersections, we should select the minimum value of the parameter. For non-zero value of p_k , we calculate $r_k = q_k / p_k$ for $k = 1, 2, 3, 4$
- For each value of k , such that $p_k < 0$, find r_k and, $u_1 = \max(0, r_k)$.
- For each value of k , such that $p_k > 0$, find r_k and, $u_2 = \min(1, r_k)$.
- Where, u_1 is the parameter value of the line where it enters in the clipping region, and u_2 is the parameter value of the line where it leaves the clipping region. If the value of u_1 is greater than u_2 , the line is completely outside. Otherwise visible segment $P'_1 P'_2$ of the line is computed as,

$$P'_1 = P_1 + (P_2 - P_1) \cdot u_1$$

$$P'_2 = P_1 + (P_2 - P_1) \cdot u_2$$

5.4.2.2 Algorithm

Algorithm LIANG_BARSKY_LINE_CLIPPING()

$t_{\min} \leftarrow 0$

```

 $t_{max} \leftarrow 1$ 
Calculate t values ( $t_{left}$ ,  $t_{right}$ ,  $t_{top}$ ,  $t_{bottom}$ )
if  $t < t_{min}$  then
    Ignore that and move to next edge
else
    Classify t as entering or leaving t
end
if t is entering then
     $t_{min} \leftarrow t$ 
if t is leaving then
     $t_{max} \leftarrow t$ 
if  $t_{min} < t_{max}$  then
    Draw a line from  $(x_1 + t_{min} * (x_2 - x_1), y_1 + t_{min} * (y_2 - y_1))$  to  $(x_1 + t_{max} * (x_2 - x_1), y_1 + t_{max} * (y_2 - y_1))$ 
if line crosses the window then
     $(x_1 + t_{min} * (x_2 - x_1), y_1 + t_{min} * (y_2 - y_1))$  and  $(x_1 + t_{max} * (x_2 - x_1), y_1 + t_{max} * (y_2 - y_1))$  are the intersection
points of line with window boundary

```

5.4.2.3 Pros and Cons

Advantages

- More efficient.
- Only requires one division to update u_1 and u_2 .
- It is more efficient than Cohen Sutherland since intersection calculation is reduced.
- It requires only one division and window intersection is computed only once.
- Works for rectangular clipping region in 2D and 3D.
- Cohen Sutherland algorithm may perform repeated computation even though the line is completely outside the window.

Disadvantage

Involves a parametric equation in computation

5.4.2.4 Example

- Ex. 5.4.5 :** Using Liang Barsky algorithm, find the clipping coordinates of the line segment with end coordinates, A (- 10, 50) and B(30, 80) against the window ($X_{min} = - 30$, $Y_{min} = 10$) ($X_{max} = 20$, $Y_{max} = 60$).



Soln. :

Here, $X_{w\min} = -30$, $Y_{w\min} = 10$, $X_{w\max} = 20$, $Y_{w\max} = 60$.

End points of line are $A = (x_1, y_1) = (-10, 50)$ and $B = (x_2, y_2) = (30, 80)$.

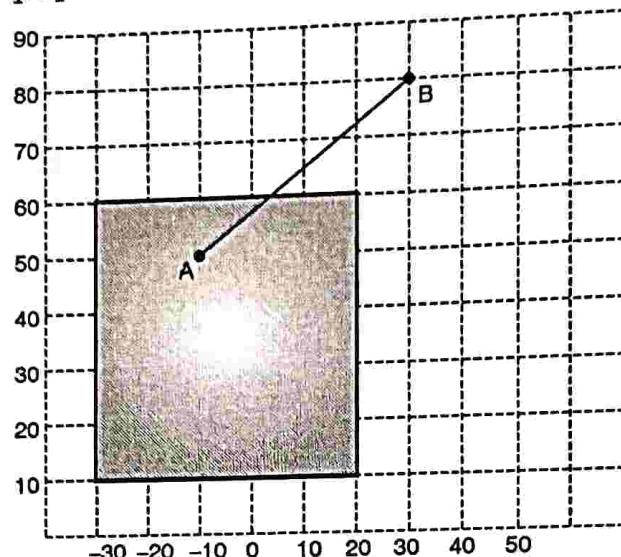


Fig. P. 5.4.5

Let us compute p_i , q_i and r_i for $i = 1, 2, 3, 4$

$$p_1 = -(\Delta x) = -(x_2 - x_1) = -(30 - (-10)) = -40$$

$$p_2 = \Delta x = 40$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(80 - 50) = -30$$

$$p_4 = \Delta y = 30$$

None of the p_i is zero, so line is neither horizontal, nor vertical. Let us compute q_i .

$$q_1 = x_1 - x_{w\min} = -10 + 30 = 20$$

$$q_2 = x_{w\max} - x_1 = 20 - (-10) = 30$$

$$q_3 = y_1 - y_{w\min} = 50 - 10 = 40$$

$$q_4 = y_{w\max} - y_1 = 60 - 50 = 10$$

$$r_1 = \frac{q_1}{p_1} = \frac{20}{-40} = -0.5$$

$$r_2 = \frac{q_2}{p_2} = \frac{30}{40} = 0.75$$

$$r_3 = \frac{q_3}{p_3} = \frac{40}{-30} = -1.33$$

$$r_4 = \frac{q_4}{p_4} = \frac{10}{30} = 0.33$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3) = \max(0, -0.5, -1.33) = 0$$

$$u_2 = \min(1, \text{all } r_i \text{ having } p_i > 0) = \min(1, r_2, r_4) = \min(1, 0.75, 0.33) = 0.33$$

$$x'_1 = x_1 + \Delta x \cdot u_1 = -10 + (40) \times (0) = -10$$

$$y'_1 = y_1 + \Delta y \cdot u_1 = 50 + (30) \times (0) = 50$$

$$x'_2 = x_1 + \Delta x \cdot u_2 = -10 + (40) \times (0.33) = -10 + 13.2 = 3.2$$

$$y'_2 = y_1 + \Delta y \cdot u_2 = 50 + (30) \times (0.33) = 50 + 9.9 = 59.9$$

\therefore Visible line segment is (-10, 50) to (3.2, 59.9)

Original line Segments	Visible line Segments
A(-10, 50) and B(30, 80)	A'(-10, 50) and B'(3.2, 59.9)

- Ex 5.4.6: Explain an algorithm which uses the parametric equation for line clipping, using the same algorithm, find the coordinates of the line segment A(10, 10), B(70, 40), after it is clipped against a window with two diagonal vertices at (20, 20) and (40, 50).

Soln.: Here, $X_{w\min} = 20$, $Y_{w\min} = 20$, $X_{w\max} = 40$, $Y_{w\max} = 50$.

End points of line are A = $(x_1, y_1) = (10, 10)$ and B = $(x_2, y_2) = (70, 40)$.

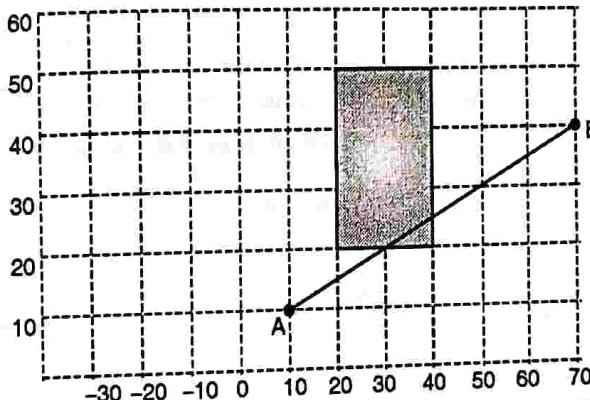


Fig. P. 5.4.6

Let us compute p_i , q_i and r_i for $i = 1, 2, 3, 4$

$$p_1 = -(\Delta x) = -(x_2 - x_1) = -(70 - 10) = -60$$

$$p_2 = \Delta x = 60$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(40 - 10) = -30$$

$$p_4 = \Delta y = 30$$

None of the p_i is zero, so line is neither horizontal, nor vertical. Let us compute q_i .

$$q_1 = x_1 - x_{w\min} = 10 - 20 = -10$$

$$q_2 = x_{w\max} - x_1 = 40 - 10 = 30$$

$$q_3 = y_1 - y_{w\min} = 10 - 20 = -10$$

$$q_4 = y_{w\max} - y_1 = 50 - 10 = 40$$

$$r_1 = \frac{q_1}{p_1} = \frac{-10}{-60} = 0.167$$



$$r_2 = \frac{q_2}{p_2} = \frac{30}{60} = 0.5$$

$$r_3 = \frac{q_3}{p_3} = \frac{-10}{-30} = 0.333$$

$$r_4 = \frac{q_4}{p_4} = \frac{40}{30} = 1.333$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3) = \max(0, 0.167, 0.333) = 0.333$$

$$u_2 = \min(1, \text{all } r_i \text{ having } P_i > 0) = \min(1, r_2, r_4) = \min(1, 0.5, 1.333) = 0.5$$

$$x'_1 = x_1 + \Delta x \cdot u_1 = 10 + (60) \times (0.333) = 10 + 19.98 = 28.98$$

$$y'_1 = y_1 + \Delta y \cdot u_1 = 10 + (30) \times (0.333) = 10 + 10 = 20$$

$$x'_2 = x_1 + \Delta x \cdot u_2 = 10 + (60) \times (0.5) = 10 + 30 = 40$$

$$y'_2 = y_1 + \Delta y \cdot u_2 = 10 + (30) \times (0.5) = 10 + 15 = 25$$

\therefore Visible line segment is (28.98, 20) to (40, 25)

Original line Segments	Visible line Segments
A(10, 10) and B(70, 40)	A' (28.98, 20) and B' (40, 25)

Ex. 5.4.7 : Clip the line using Liang Barsky algorithm against the window with $(X_{w_{\min}}, Y_{w_{\min}}) = (0,0)$ and $(X_{w_{\max}}, Y_{w_{\max}}) = (100, 50)$. Line endpoints are A(10, 10) and B(110, 40).

Soln. :

Here, $X_{w_{\min}} = 0$, $Y_{w_{\min}} = 0$, $X_{w_{\max}} = 100$, $Y_{w_{\max}} = 50$.

End points of line are A = $(x_1, y_1) = (10, 10)$ and B = $(x_2, y_2) = (110, 40)$.

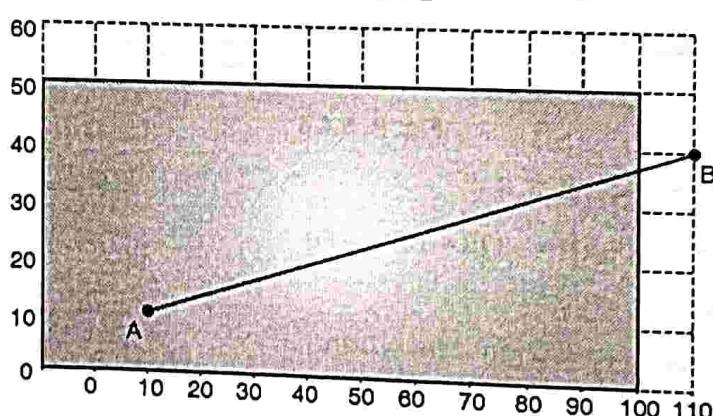


Fig. P. 5.4.7

Let us compute p_i , q_i and r_i for $i = 1, 2, 3, 4$

$$p_1 = -(\Delta x) = -(x_2 - x_1) = -(110 - 10) = -100$$

$$p_2 = \Delta x = 100$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(40 - 10) = -30$$

$$p_4 = \Delta y = 30$$

None of the p_i is zero, so line is neither horizontal, nor vertical. Let us compute q_i .

$$q_1 = x_1 - x_{w\min} = 10 - 0 = 10$$

$$q_2 = x_{w\max} - x_1 = 100 - 10 = 90$$

$$q_3 = y_1 - y_{w\min} = 10 - 0 = 10$$

$$q_4 = y_{w\max} - y_1 = 50 - 10 = 40$$

$$r_1 = \frac{q_1}{p_1} = -\frac{10}{100} = -0.1$$

$$r_2 = \frac{q_2}{p_2} = \frac{90}{100} = 0.9$$

$$r_3 = \frac{q_3}{p_3} = \frac{10}{-30} = -0.33$$

$$r_4 = \frac{q_4}{p_4} = \frac{40}{30} = 1.33$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3) = \max(0, -0.1, -0.33) = 0$$

$$u_2 = \min(1, \text{all } r_i \text{ having } P_i > 0) = \min(1, r_2, r_4) = \min(1, 0.9, 1.33) = 0.9$$

$$u_1 = 0; \text{ So, } (x'_1, y'_1) = (x_1, y_1)$$

$$x'_2 = x_1 + \Delta x \cdot u_2 = 10 + (100) \times (0.9) = 10 + 90 = 100$$

$$y'_2 = y_1 + \Delta y \cdot u_2 = 10 + (30) \times (0.9) = 10 + 27 = 37$$

\therefore Visible line segment is (10, 10) to (100, 37)

Original line Segments	Visible line Segments
A(10, 10) and B(110, 40)	A'(10, 10) and B'(100, 37)

E.5.4.8: Determine the visible span of line AB against given clipping window using Liang Barsky algorithm.

Soln.:

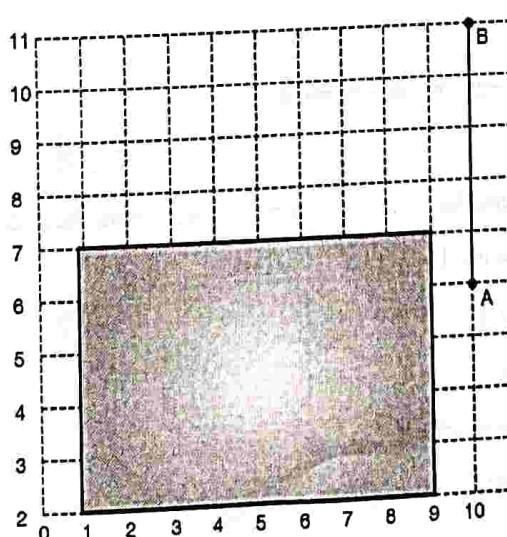


Fig. P. 5.4.8



Let us consider line end points $(x_1, y_1) = (11, 6)$ and $(x_2, y_2) = (11, 10)$. From clipping window position, $X_{w\min} = 1$, $X_{w\max} = 9$, $Y_{w\min} = 2$ and $Y_{w\max} = 8$. Let us compute p_i , q_i and r_i ,

$$P_1 = -\Delta x = -(x_2 - x_1) = -(11 - 11) = 0$$

$$P_2 = \Delta x = (x_2 - x_1) = 11 - 11 = 0$$

$$q_1 = x_1 - X_{w\min} = 11 - 1 = 10$$

$$q_2 = X_{w\max} - x_1 = 9 - 11 = -2$$

Here, $p_1 = 0$ so line is vertical, and $q_1 < 0$ implies that line is right side of the right boundary of clipping window. So line is trivially rejected.

Ex. 5.4.9 : Determine the visible span of line AB against given clipping window using Liang Barsky algorithm.

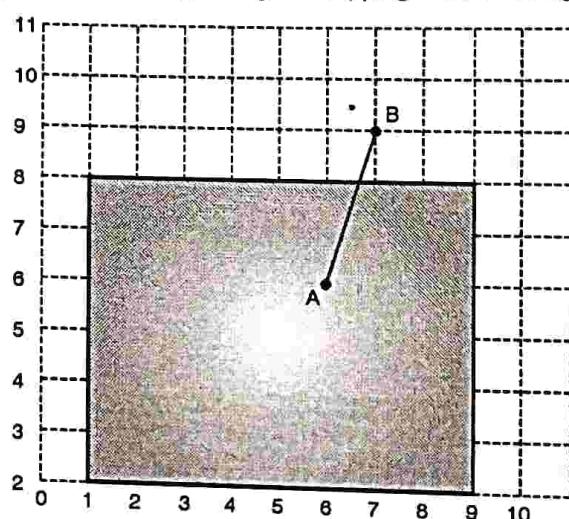


Fig. P. 5.4.9

Soln. :

Let us consider line end points $(x_1, y_1) = (6, 6)$ and $(x_2, y_2) = (8, 9)$ and From clipping window, $X_{w\min} = 1$, $X_{w\max} = 9$, $Y_{w\min} = 2$ and $Y_{w\max} = 8$. Let us compute p_i , q_i and r_i .

$$P_1 = -\Delta x = -(x_2 - x_1) = -(8 - 6) = -2$$

$$P_2 = \Delta x = 2$$

$$P_3 = -\Delta y = -(y_2 - y_1) = -(9 - 6) = -3$$

$$P_4 = \Delta y = 3$$

None of the p_i is zero, so line is neither horizontal nor vertical. Let us check if it passes through window.

$$q_1 = x_1 - X_{w\min} = 6 - 1 = 5$$

$$q_2 = X_{w\max} - x_1 = 9 - 6 = 3$$

$$q_3 = y_1 - Y_{w\min} = 6 - 2 = 4$$

$$q_4 = Y_{w\max} - y_1 = 8 - 6 = 2$$

We compute r_i as follow to find intersection of line with window boundary.

$$r_i = \frac{q_i}{p_i}, i = 1, 2, 3, 4$$

$$r_1 = \frac{q_1}{p_1} = -\frac{5}{2} = -2.5$$

$$r_2 = \frac{q_2}{p_2} = \frac{3}{2} = 1.5$$

$$r_3 = \frac{q_3}{p_3} = -\frac{4}{3} = -1.33$$

$$r_4 = \frac{q_4}{p_4} = \frac{2}{3} = 0.66$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3) = \max(0, -2.5, -1.33) = 0$$

$$u_2 = \min(1, \text{all } r_i \text{ having } p_i > 0) = \min(1, r_2, r_4) = \min(1, 1.5, 0.66) = 0.66$$

Let us consider (x'_1, y'_1) and (x'_2, y'_2) are the end points of clipped line. From parametric equation of line,

$$5 = y_1 + \Delta y \cdot u_2 = 6 + 3 \times 0.66 = 8$$

Visible line segment is (6, 6) to (7.33, 8)

Original line Segments	Visible line Segments
A(6, 6) and B(8, 9)	A'(6, 6) and B'(7.33, 8)

Q. 5.4.10: Explain Liang Barsky line clipping algorithm. Apply the algorithm to clip the line with coordinates (30, 60) and (60, 20) against window $(x_{w_{\min}}, y_{w_{\min}}) = (10, 10)$ and $(x_{w_{\max}}, y_{w_{\max}}) = (50, 50)$. MU - May 18, May 19, 10 Marks

Soln.:

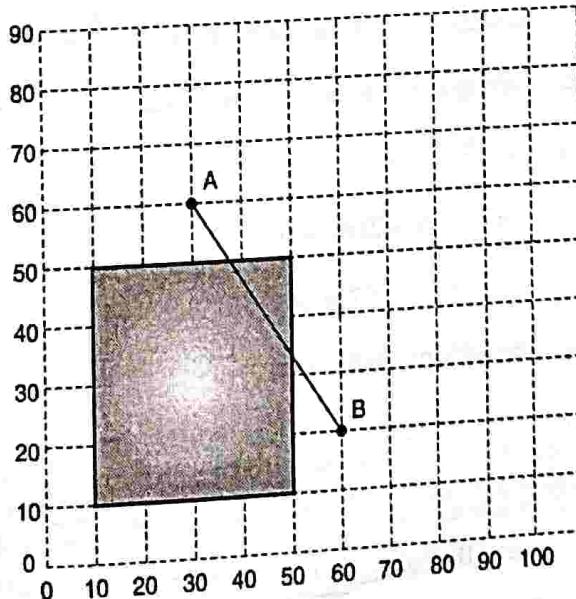


Fig. P. 5.4.10

Here, $X_{w_{\min}} = 10$, $Y_{w_{\min}} = 10$, $X_{w_{\max}} = 50$, $Y_{w_{\max}} = 50$.



End points of line are $A = (x_1, y_1) = (30, 60)$ and $B = (x_2, y_2) = (60, 20)$.

Let us compute p_i, q_i and r_i for $i = 1, 2, 3, 4$

$$p_1 = -(\Delta x) = -(x_2 - x_1) = -(60 - 30) = -30$$

$$p_2 = \Delta x = 30$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(20 - 60) = 40$$

$$p_4 = \Delta y = -40$$

None of the p_i is zero, so line is neither horizontal, nor vertical. Let us compute q_i .

$$q_1 = x_1 - X_{w_{min}} = 30 - 10 = 20$$

$$q_2 = X_{w_{max}} - x_1 = 50 - 30 = 20$$

$$q_3 = y_1 - Y_{w_{min}} = 60 - 10 = 50$$

$$q_4 = Y_{w_{max}} - y_1 = 50 - 60 = -10$$

$$r_1 = \frac{q_1}{p_1} = \frac{20}{-30} = -0.67$$

$$r_2 = \frac{q_2}{p_2} = \frac{20}{30} = 0.67$$

$$r_3 = \frac{q_3}{p_3} = \frac{50}{40} = 1.25$$

$$r_4 = \frac{q_4}{p_4} = \frac{-10}{-40} = 0.25$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_4) = \max(0, -0.67, 0.25) = 0.25$$

$$u_2 = \min(1, \text{all } r_i \text{ having } P_i > 0) = \min(1, r_2, r_3) = \min(1, 0.67, 1.25) = 0.67$$

$$x'_1 = x_1 + \Delta x \cdot u_1 = 30 + (30) \times (0.25) = 37.5$$

$$y'_1 = y_1 + \Delta y \cdot u_1 = 60 + (-40) \times (0.25) = 50$$

$$x'_2 = x_1 + \Delta x \cdot u_2 = 30 + (30) \times (0.67) = 50$$

$$y'_2 = y_1 + \Delta y \cdot u_2 = 60 + (-40) \times (0.67) = 33.2$$

\therefore Visible line segment is (37.5, 50) to (50, 33.2)

Original Line Segments	Visible line Segments
A(30, 60) and B(60, 20)	A'(37.5, 50) and B'(50, 33.2)

Ex. 5.4.11 : Explain Liang Barsky line clipping algorithm. Apply this algorithm to the line 10 with coordinates (35, 60) and (80, 25) against the window $(X_{w_{min}}, Y_{w_{min}}) = (10, 10)$ and $(X_{w_{max}}, Y_{w_{max}}) = (50, 50)$. MU - Dec. 18, 10 Marks

Soln.:

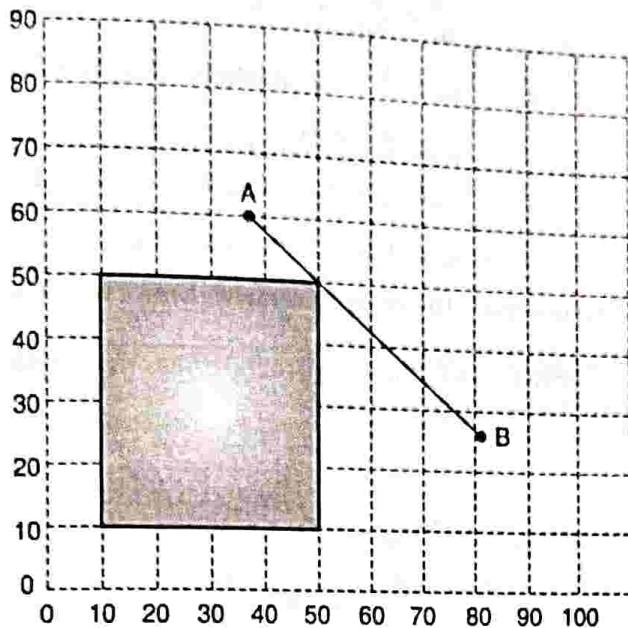


Fig. P. 5.4.11

Let us consider line end points $(x_1, y_1) = (35, 60)$ and $(x_2, y_2) = (80, 25)$ and From clipping window, $x_{w\min} = 10$, $x_{w\max} = 50$, $y_{w\min} = 10$ and $y_{w\max} = 50$. Let us compute p_i , q_i and r_i .

$$p_1 = -\Delta x = -(x_2 - x_1) = -(80 - 35) = -45$$

$$p_2 = \Delta x = 45$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(25 - 60) = -35$$

$$p_4 = \Delta y = 35$$

None of the p_i is zero, so line is neither horizontal nor vertical. Let us check if it passes through window.

$$q_1 = x_1 - X_{w\min} = 35 - 10 = 25$$

$$q_2 = X_{w\max} - x_1 = 50 - 35 = 15$$

$$q_3 = y_1 - Y_{w\min} = 60 - 10 = 50$$

$$q_4 = Y_{w\max} - y_1 = 50 - 60 = -10$$

We compute r_i as follow to find intersection of line with window boundary.

$$R_i = \frac{q_i}{p_i}, i = 1, 2, 3, 4$$

$$r_1 = \frac{q_1}{p_1} = \frac{25}{-45} = -0.55$$

$$r_2 = \frac{q_2}{p_2} = \frac{-15}{45} = 0.333$$

$$r_3 = \frac{q_3}{p_3} = \frac{50}{-35} = -1.43$$

$$r_4 = \frac{q_4}{p_4} = \frac{-10}{35} = -0.29$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3)$$

$$= \max(0, -0.55, -1.43) = 0$$

$$u_2 = \min(1, \text{all } r_i \text{ having } p_i > 0) = \min(1, r_2, r_4)$$

$$= \min(1, 0.33, -0.29) = -0.29$$

Here, $u_1 > u_2$, so line is completely outside the clipping window. Entire line will be clipped.

Ex. 5.4.12 : Clip the line with co-ordinates $(5, 10)$ and $(35, 30)$ against the window $(x_{\min}, y_{\min}) = (10, 10)$ and $(x_{\max}, y_{\max}) = (20, 20)$.

MU - Dec. 19, 5 Marks

Soln. :

Here, $X_{w_{\min}} = 10$, $Y_{w_{\min}} = 10$, $X_{w_{\max}} = 20$, $Y_{w_{\max}} = 20$.

End points of line are $A = (x_1, y_1) = (5, 10)$ and $B = (x_2, y_2) = (35, 30)$.

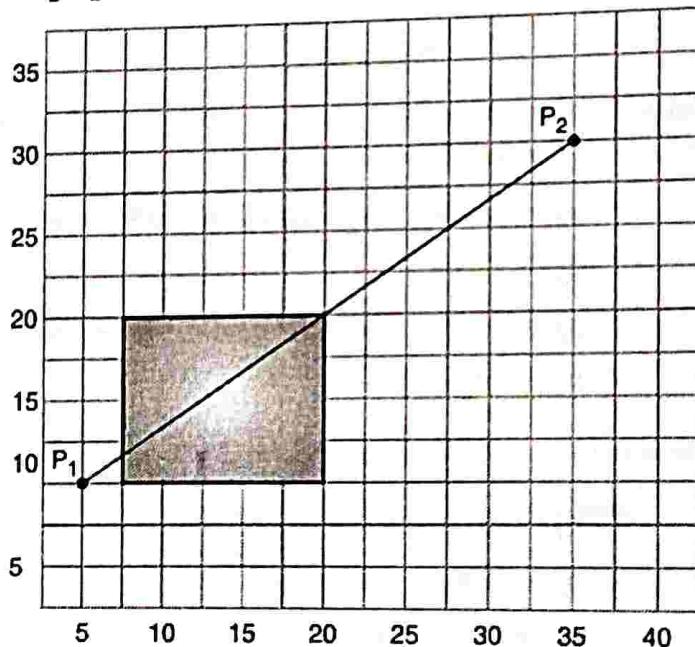


Fig. P.5.4.12

Let us compute p_i , q_i and r_i for $i = 1, 2, 3, 4$

$$p_1 = -(\Delta x) = -(x_2 - x_1) = -(35 - 5) = -30$$

$$p_2 = \Delta x = 30$$

$$p_3 = -\Delta y = -(y_2 - y_1) = -(30 - 10) = -20$$

$$p_4 = \Delta y = 20$$

None of the p_i is zero, so line is neither horizontal, nor vertical. Let us compute q_i .

$$q_1 = x_1 - x_{w_{\min}} = 5 - 10 = -5$$

$$q_2 = x_{w_{\max}} - x_1 = 20 - 5 = 15$$

$$q_3 = y_1 - y_{w_{\min}} = 10 - 10 = 0$$

$$q_4 = y_{w_{\max}} - y_1 = 20 - 10 = 10$$

$$r_1 = \frac{q_1}{p_1} = \frac{-5}{-30} = 0.17$$

$$r_2 = \frac{q_2}{p_2} = \frac{15}{30} = 0.5$$

$$r_3 = \frac{q_3}{p_3} = 0$$

$$r_4 = \frac{q_4}{p_4} = \frac{10}{20} = 0.5$$

$$u_1 = \max(0, \text{all } r_i \text{ having } p_i < 0) = \max(0, r_1, r_3) = \max(0, 0.17, 0) = 0.1667$$

$$u_2 = \min(1, \text{all } r_i \text{ having } P_i > 0) = \min(1, r_2, r_4) = \min(1, 0.5, 0.5) = 0.5$$

$$x'_1 = x_1 + \Delta x \cdot u_1 = 5 + (30) \times (0.1667) = 35$$

$$y'_1 = y_1 + \Delta y \cdot u_1 = 10 + (20) \times (0.1667) = 13.33$$

$$x'_2 = x_1 + \Delta x \cdot u_2 = 5 + (30) \times (0.5) = 5 + 15 = 20$$

$$y'_2 = y_1 + \Delta y \cdot u_2 = 10 + (20) \times (0.5) = 10 + 10 = 20$$

\therefore Visible line segment is (35, 13.33) to (20, 20)

Original line Segments	Visible line Segments
A(5, 10) and B(35, 30)	A'(35, 13.33) and B'(20, 20)

5.5 Polygon Clipping

- Clipping of polygon requires more attention.
- The output of line clipping is either point or line.
- The output of polygon clipping may be point, line or polygon.
- Following issues arise in polygon clipping. All in all, polygon clipping is more complex compared to line clipping.
- In polygon clipping, each edge needs to be tested against each edge of the clipping region. This process may add new edges, may delete existing edges, may retain or divide the edges.

Issues in Polygon Clipping

1. Clipping a line segment yields at the most one-line segment.
2. Clipping of convex polygon yields at most one polygon (Fig. 5.5.1(a)).
3. Clipping of concave polygon can yield multiple polygons (Fig. 5.5.1(b)).
4. For the same polygon, the different number of edges may be generated depending on its orientation and spatial location (Fig. 5.5.1(c)).
5. May add new edges.
6. Edge may be discarded or divided.

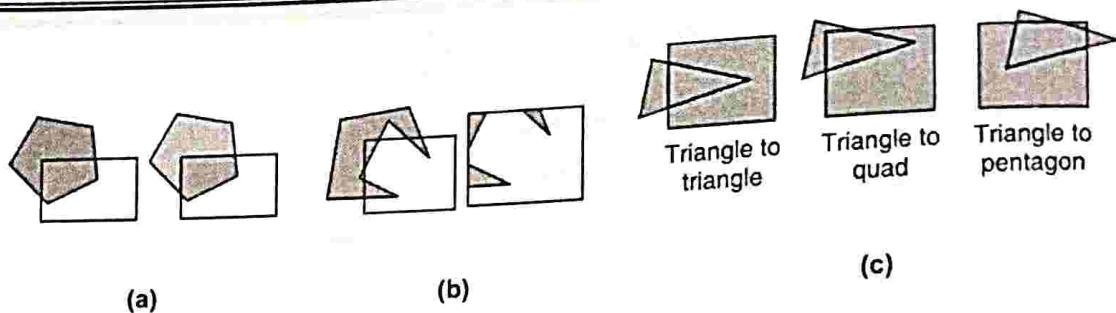


Fig. 5.5.1 : Cases of polygon clipping

5.5.1 Sutherland-Hodgeman Polygon Clipping Algorithm

Q. Explain the Sutherland-Hodgeman polygon clipping algorithm with suitable example and comment on its short coming
MU - May 18, 10 Marks

Q. Explain any one Polygon clipping algorithm.
MU - Dec. 18, 10 Marks

5.5.1.1 Working Mechanism

- Sutherland-Hodgeman polygon clipping algorithm uses a divide-and-conquer strategy. Divide and conquer method divides the large problem into small subproblems, solve them recursively and combine the results to build the solution of parent problem.
- Sutherland-Hodgeman algorithm works in four stages. In each stage, each edge of subject polygon is tested against the edge of the clipping rectangle.
- To accomplish this task, the algorithm process all vertices against each clip rectangle boundary in turn. Beginning with the initial set of polygon vertices, we could first clip the polygon against any of the edge (ie left, right, top or bottom) to produce a new sequence of vertices.
- This new set of vertices then successively passed to remaining boundary clippers. At each step, a new sequence of output vertices is generated and passed to the next clipper. Fig. 5.5.2 shows the simulation of the algorithm.

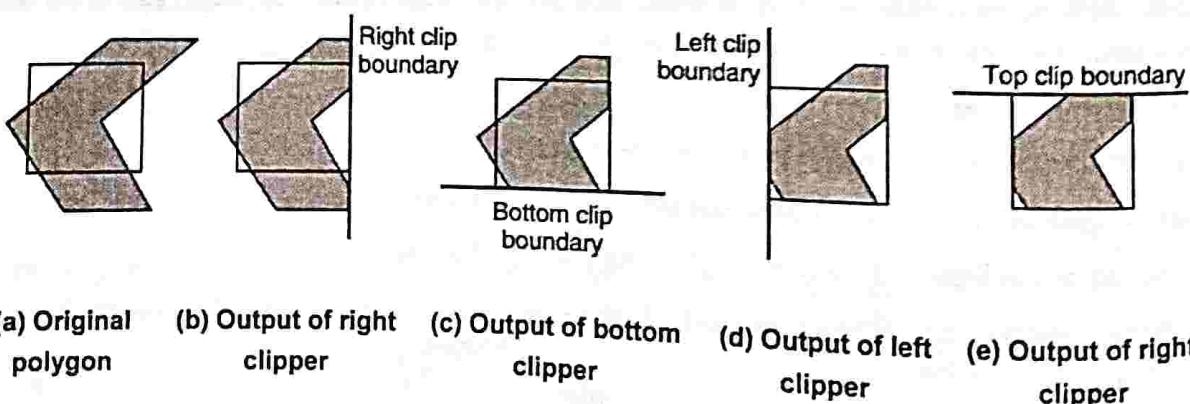


Fig. 5.5.2 : Stages of polygon clipping

When we traverse along the polygon boundary, one of the following four cases occurs :

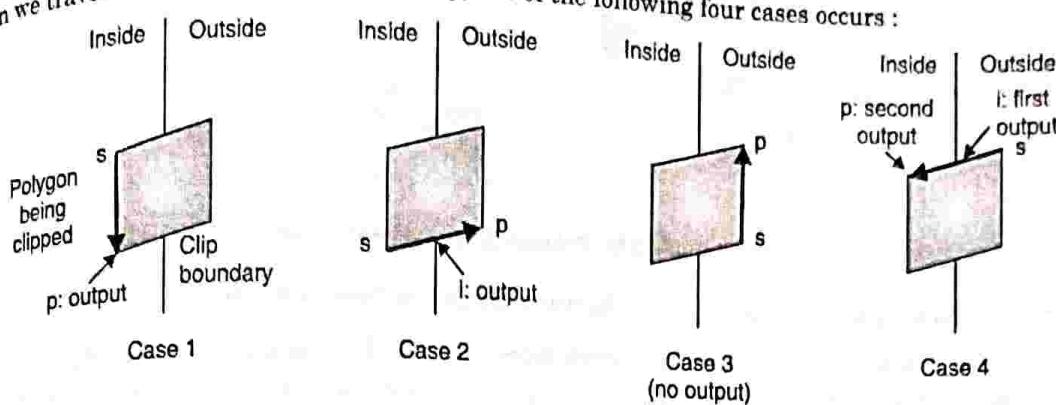


Fig. 5.5.3 : Output for various clippers

Case 1 : While traversing along the boundary of the subject polygon, if the movement is from inside to inside i.e. when the source(s) and destination(p) vertices of the edge are inside the clipping region, we output destination vertex (p) only.

Case 2 : When the movement is from inside to outside i.e. when source vertex is inside the clipping region and destination vertex is outside the clipping region, the polygon edge intersects the clipping boundary. In that case, we output the intersection point 'i' only.

Case 3 : When the movement is from outside to outside i.e when source vertex is outside the clipping region and destination vertex is also outside the clipping region, nothing is selected for output.

Case 4 : When the movement is from outside to inside i.e. when source vertex is outside the clipping region and destination vertex is inside the clipping region, the polygon edge intersects the clipping boundary. In that case, we output both, the intersection point and the destination vertex.

Output list of the first clipper is passed to next clipper and processed in the same way.

Convex polygons are correctly clipped by this algorithm, but concave polygons sometimes may generate extraneous lines as shown in Fig. 5.5.4. Edge ab was not part of original polygon, but it has been now part of output polygon.

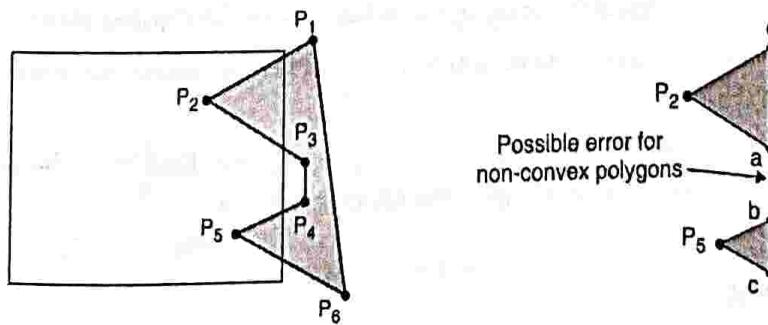


Fig. 5.5.4 : Issue with the Sutherland-Hodgeman algorithm

This problem can be handled in many ways. One way is to convert the concave polygon to multiple convex polygons as shown in Fig. 5.5.5. The process of converting a polygon to a set of triangles is known as **tessellation**. Then process each convex polygon individually.

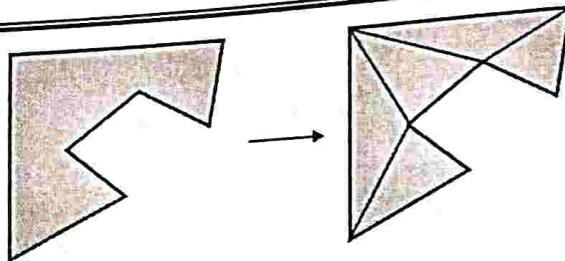


Fig. 5.5.5 : Tessellation process

- Another way of solving this problem is to tag the different new vertices as follow.
- Call α the new vertex generated on the transition from IN to OUT. Call β the new vertex generated on the transition from OUT to IN. Starting from OUT, connect α and β as soon as is generated. This process successfully eliminates the extraneous edge.
- This algorithm is well suited for hardware implementation and can easily be parallelized on multiple processors, each processor works as a clipper. However, this method works only for convex clipping region.

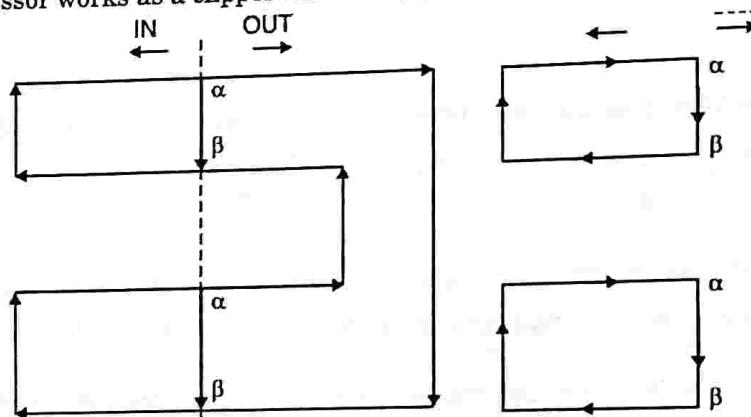


Fig. 5.5.6 : Solving the issue of Sutherland-Hodgeman algorithm

5.5.1.2 Algorithm

1. Read coordinates of vertices of the Polygon.
2. Read coordinates of the clipping window
3. Consider the left edge of the window
4. Compare the vertices of each edge of the polygon, individually with the clipping plane.
5. Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary.
6. Repeat the steps 4 and 5 for remaining edges or the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.
7. Stop.

5.5.1.3 Pros and Cons

Advantages

- It is easy to implement
- It clips polygon against all edges of the clipping polygon
- Well suited for hardware implementation

Disadvantages

- Only works for convex clipping polygon
- It clips to each window boundary one at a time
- It has redundant edge-line cross computation
- Requires a considerable amount of memory to store all polygons in the original form
- Sometimes produces single polygon instead of multiple polygons for concave polygon

5.5.1.4 Examples

Ex. 5.5.1 : Clip the subject polygon ($V_1 - V_2 - V_3 - V_4 - V_5 - V_1$) against the given clipping polygon.

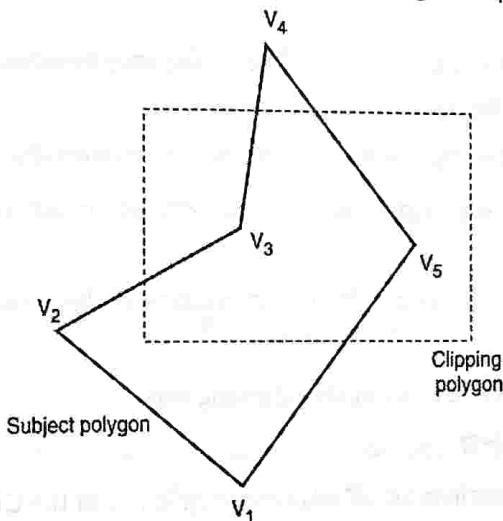
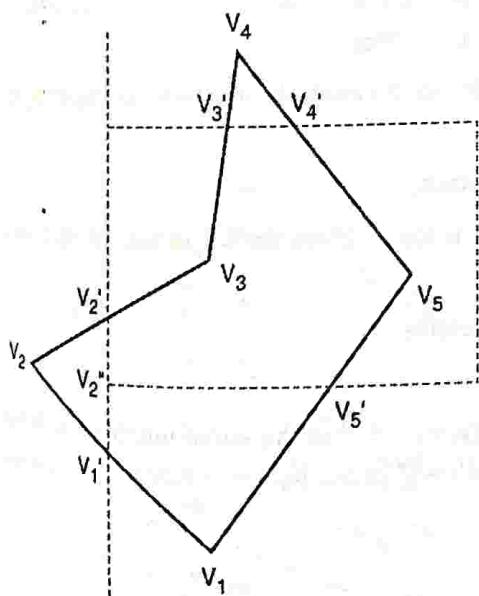


Fig. P. 5.5.1

Soln. : The intersection of the different clippers with subject polygon is shown in the following Fig. P. 5.5.1 :



Input to the algorithm : $\{V_1, V_2, V_3, V_4, V_5\}$

The output of left clipper : $\{V_1, V_1', V_2', V_3, V_4, V_5\}$

The output of right clipper : $\{V_1, V_1', V_2', V_3, V_4, V_5\}$

The output of top clipper : $\{V_1, V_1', V_2', V_3, V_3', V_4', V_5\}$

The output of bottom clipper : $\{V_2'', V_2', V_3, V_3', V_4', V_5, V_5'\}$

Fig. P. 5.5.1(a)

The final clipped polygon would be $\{V_2'', V_2', V_3, V_3', V_4', V_5, V_5'\}$.

5.5.2 Weiler - Atherton Polygon Clipping Algorithm

MU - Dec. 18, 10 Marks

MU - May 19, 10 Marks

- Q. Explain any one Polygon clipping algorithm.
 Q. Explain Weiler Atherton polygon clipping algorithm with suitable example.

5.5.2.1 Working Mechanism

- The Weiler-Atherton clipping algorithm is a general-purpose 2D clipping algorithm. It is capable of clipping a concave polygon with interior holes to the boundaries of another concave polygon, also with interior holes. The polygon to be clipped is called the Subject Polygon (SP) and the clipping region is called the Clip Polygon (CP).
- SP and the CP are defined by a circular list of vertices. Exterior boundaries are described in clockwise, and the interior boundaries or holes are described in anti-clockwise. The boundaries of the SP and the CP may or may not intersect.
- If they intersect, the intersections occur in pairs. One of the intersections occurs when the SP edge enters inside of the CP and one when it leaves.
- Proceeding in clockwise around the original polygon vertex list, perform the following steps :
- o If a line segment enters the clip region, add the intersection to the output list and follow the polygon boundary.
- o If a line segment leaves the clip region, add the intersection to the output list, follow the boundary of the clip region, clockwise.

This could be achieved more fundamentally in the following way :

1. Creating Intersection Vertex Lists

- Entering list contains intersections for SP edge entering inside of the CP.
- Leaving list contains intersections for the SP edge leaving the inside of the CP.

2. Perform Actual Clipping

- From entering a list, remove an intersection vertex and check if the list is empty. If it is so, the clipping process is done.
- Traverse the SP vertex list until next intersection point is found. Store the SP list upto this point to the inside holding list.
- Jump to the CP vertex list at the same intersection point location.
- Traverse the CP vertex list until the next intersection point is found. Store the CP vertex list up to this point to the inside holding list.
- Jump back to SP vertex list at the same intersection point location.
- Repeat the process until the loop is formed.
- Polygons outside the CP are found using the same procedure, except that the initial intersection vertex is obtained from the leaving list and the CP vertex list is followed in the reverse direction. The polygon lists are copied to the outside holding list.

5.5.2.2 Pros and Cons

Advantages

- This algorithm can also handle concave polygons easily.
- It does not require splitting the concave polygon to convert it into a convex polygon.

Unlike the Sutherland-Hodgeman algorithm, it does not produce spurious edge, so no post-processing is required.

Disadvantage

Not easy to parallelize

5.5.2.3 Examples

Ex. 5.5.2: Consider the following pair of clipping and subject polygon. Find the visible region of the subject polygon.

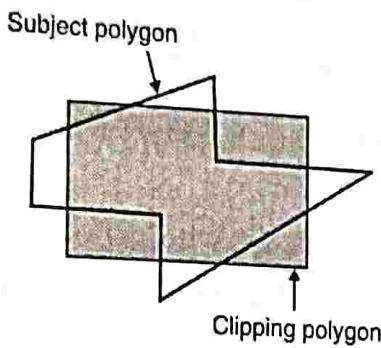


Fig. P. 5.5.2

Soln.:

Let us first label the vertices of the subject polygon as s_i and clipping polygon as c_i . Intersection points of both polygons are labelled as i_j .

As per algorithm, start visiting the polygon from any of the entering intersection points. List of entering intersection points : { i_1, i_3, i_5, i_7 }.

Let us start from i_1 and keep traversing along the window boundary as mentioned in the above procedure.

Fig. P. 5.5.2(b) shows the procedure.

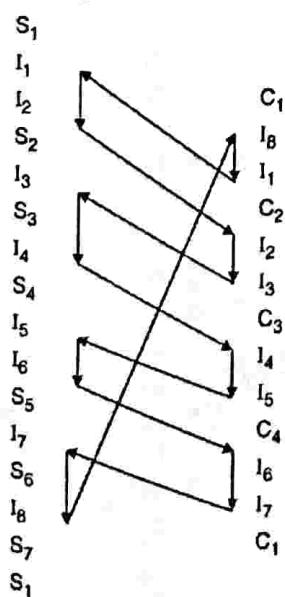


Fig. P. 5.5.2 (a)

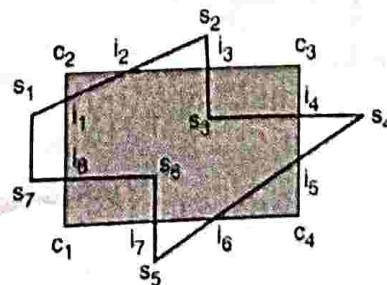


Fig. P. 5.5.2 (b)



Collect all the visited vertices in the same order. That would be the visible section of the subject polygon. So visible region of S is : { $i_1, i_2, i_3, s_3, i_4, i_5, i_6, i_7, s_6, i_8, i_1$ }

Ex. 5.5.3 : Consider the following pair of clipping and subject polygon. Find the visible region of the subject polygon.

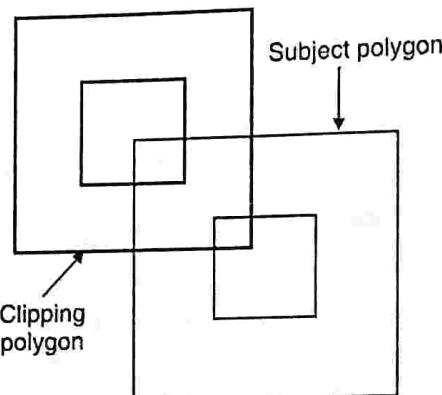


Fig. P. 5.5.3

Soln. :

Let us first label the vertices of the subject polygon as s_i and clipping polygon as c_i . Intersection points of both polygons are labelled as i_i .

As per algorithm, start visiting the polygon from any of the entering intersection points. List of entering intersection points : { i_1, i_3, i_5 }.

Let us start from i_1 and keep traversing along the window boundary as mentioned in the above procedure. Fig. P. 4.3.6(a) shows the procedure.

Collect all the visited vertices in the same order. That would be the visible section of the subject polygon.

So visible region of S is : { $i_1, i_2, c_6, i_3, i_4, i_5, s_8, i_6, i_1$ }

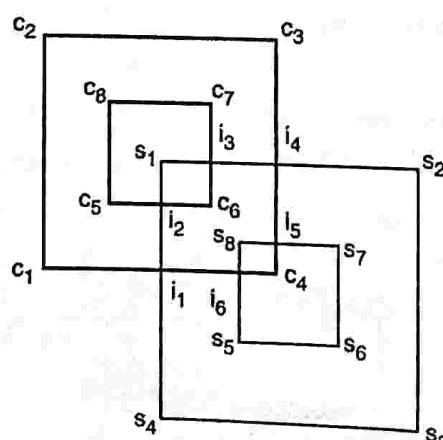


Fig. P. 5.5.3 (a)

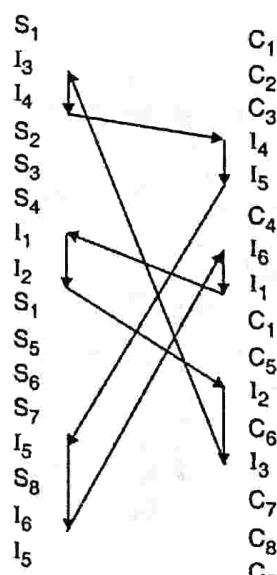


Fig. P. 5.5.3 (b)

Review Questions

- Q.1 Explain the concept of windowing and clipping.
- Q.2 Explain various approaches for clipping the primitives.
- Q.3 Write a short note on viewing pipeline.
- Q.4 Define the terms - Window, Viewport, Window-to-viewport transformation.
- Q.5 Discuss the viewing pipeline in computer graphics with a neat diagram.
- Q.6 Discuss the effect of different size and position of window and viewport in viewing.
- Q.7 Derive transformation matrix for window-to-viewport transformation.
- Q.8 What is the point clipping? How to achieve it?
- Q.9 What is line clipping? Discuss various cases of clipping with suitable diagram.
- Q.10 Write and explain with an example Cohen-Sutherland line clipping algorithm.
- Q.11 How region codes are computed in the Cohen-Sutherland Algorithm?
- Q.12 State the advantages and disadvantages of the Cohen-Sutherland Algorithm.
- Q.13 State the issues in polygon clipping.
- Q.14 Explain Sutherland-Hodgeman polygon clipping algorithm.
- Q.15 What are the issues involved in the Sutherland-Hodgeman polygon clipping algorithm? How to handle them.





3D Transformation

Syllabus

Introduction, Translation, Rotation, Scaling, Reflection, Composite transformations, Rotation about an arbitrary axis

6.1 3D Transformation

6.1.1 Introduction

- 3D transformation is an extension of 2D transformation in space.
- In 3D transformation, transformation operations like translation, rotation, scaling etc. take place in space rather than on a plane.
- Like 2D transformation, we will make use of homogenous representation to describe the 3D transformation matrices.

6.1.2 3D Geometry

- In 2D geometry, the point is described by (x, y) coordinates in XY plane, whereas in 3D geometry point (x, y, z) is defined in space with added third dimension z.
- 2D geometry system is described by the intersection of two planes which are perpendicular to each other.
- 3D geometry system is described by the intersection of three planes which are perpendicular to each other.
- Intersecting lines form the axis and the intersection point of the plane forms the origin of the geometry system.
- 2D and 3D geometry system are depicted in Fig. 6.1.1.

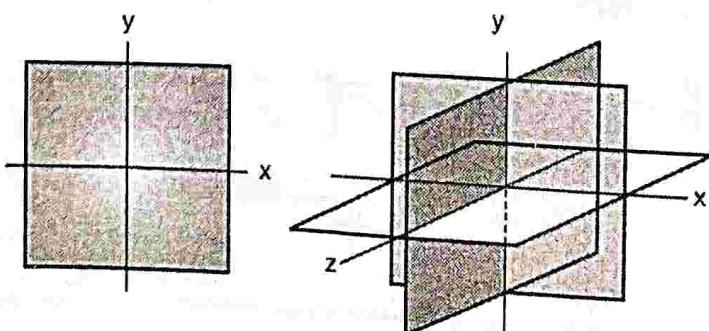


Fig. 6.1.1 : 2D geometry system (left) and 3D geometry system (right)

- 2D geometry divides the coordinate system into four quadrants. 3D geometry divides the coordinate system into eight octants.
- The distance of point is measured from the origin.

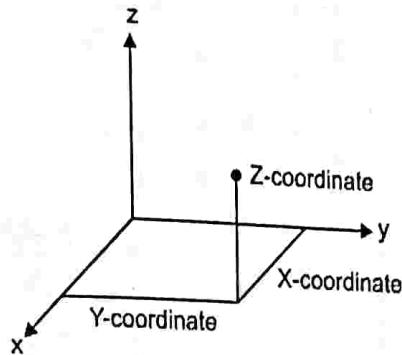


Fig. 6.1.2

Distance between two points $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ is computed by the following equation :

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

When the point is projected on any plane using a perpendicular projection, the third coordinate becomes zero. Any point on the XY plane will be $(x, y, 0)$ and any point on X-axis will be $(x, 0, 0)$. Similar observations can be made for other planes and axes.

6.2 Translation

- The **three-dimensional translation** is a process of moving an object from one location to another location along the *straight path in space*.
- The translation is achieved by adding the required amount of shift in each direction.
- Amount of translation is specified by *translation vector* or *shift vector*, $T = [t_x, t_y, t_z]$.
- Let us consider the original point $P (x, y, z)$, which becomes $P' (x', y', z')$ after translation. Translated coordinates are computed by following equations :

$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

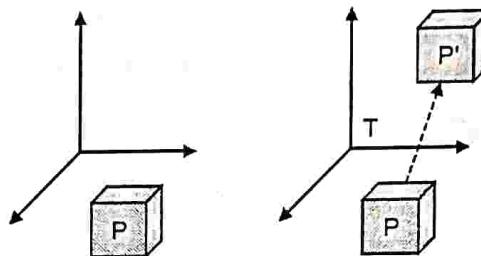
Equivalent homogeneous matrix representation is :

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

Inverse Translation Matrix

We can achieve inverse translation by negating the shift parameters. Inverse translation matrix is given as,

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



(a) Original object P (b) Translated object P'

Fig. 6.2.1 : Translation of object in space

Ex. 6.2.1 : Translate a triangle with vertices A (2, 2, 2), B (3, 4, 7) and C (8, 9, 12) by translation vector T [2, 4, 5].

Soln. :

Here translation vector $T = [2, 4, 5]$. Let us consider $P = [A, B, C]$ is a set of original points and $P' = [A', B', C']$ is the set of transformed coordinates. Transformed coordinates are computed as,

$$P' = T \cdot P = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 10 \\ 6 & 8 & 13 \\ 9 & 12 & 17 \\ 1 & 1 & 1 \end{bmatrix}$$

Translated vertices are $A' (4, 6, 9)$, $B' (5, 8, 12)$ and $C' (10, 13, 17)$.

Original Coordinates	Transformed Coordinates
A(2, 2, 4)	A' (4, 6, 9)
B(3, 4, 7)	B' (5, 8, 12)
C(8, 9, 12)	C' (10, 13, 17)

6.3 Scaling

- 3D scaling is performed by multiplying all coordinates of the object by some constants called scaling parameters $S = [S_x, S_y, S_z]$. Scaling alters the shape and size of the object; hence it is not rigid-body transformation.
- Scaling can be performed in either of the following ways :
 1. Scaling with respect to the origin

2. Scaling with respect to the reference point

6.3.1 Scaling with Respect to Origin

Let, $S = [S_x, S_y, S_z]$ be a vector of the scaling parameters in all three directions, scaling with respect to the origin of point P is computed as,

$$P' = S \cdot P$$

$$x' = S_x \cdot x$$

$$y' = S_y \cdot y$$

$$z' = S_z \cdot z$$

Matrix Representation of Scaling Transformation

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

If all three parameters are the same, it is called **uniform scaling**, which preserves the original shape of the object to be scaled. Otherwise, it is called **non-uniform scaling**.

Fig. 6.3.1 shows the scaling operation with respect to the origin.

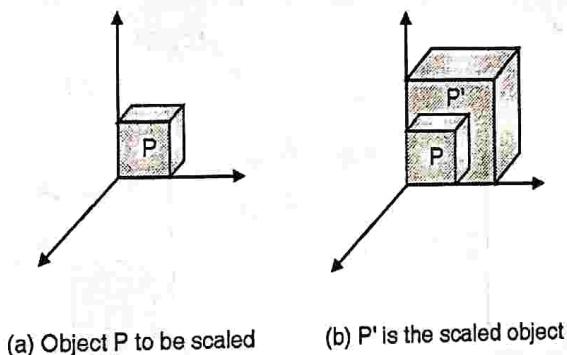


Fig. 6.3.1 : 3D Scaling with respect to the origin

6.3.2 Scaling with Respect to Reference Point

Scaling with respect to some reference point (x_r, y_r, z_r) is carried out by the following sequence of the steps :

- Translate reference point to the origin.
- Apply scaling transformation with respect to the origin.
- Perform inverse translation to move the reference point back to the original position.
- The sequence of operations to find transformed coordinates is given as,

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

more specifically

$$P' = T_{(x_r, y_r, z_r)} \cdot S_{(S_x, S_y, S_z)} \cdot T_{(-x_r, -y_r, -z_r)} \cdot P$$

- Matrix representation of the operation would be,

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & (1-S_x)x_r \\ 0 & S_y & 0 & (1-S_y)y_r \\ 0 & 0 & S_z & (1-S_z)z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Inverse scaling is achieved by replacing scaling parameters with their reciprocal.
- Fig. 6.3.2 depicts the scaling operation with respect to the reference point.

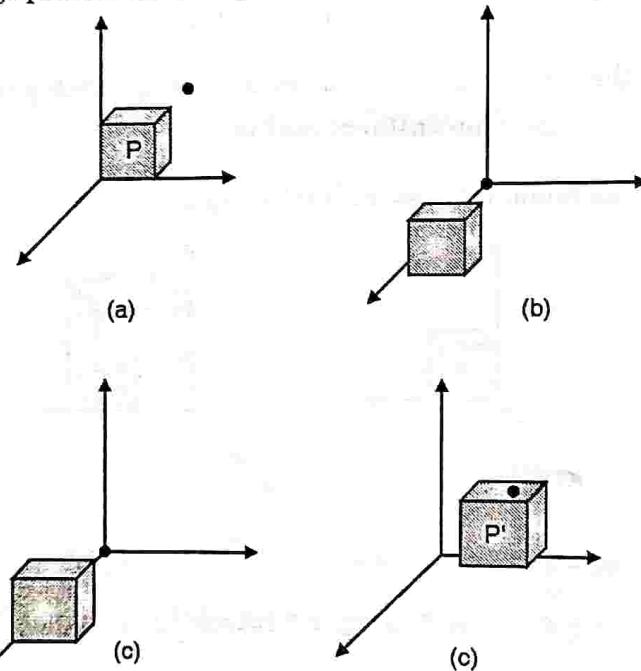


Fig. 6.3.2 : 3D scaling with respect to a reference point

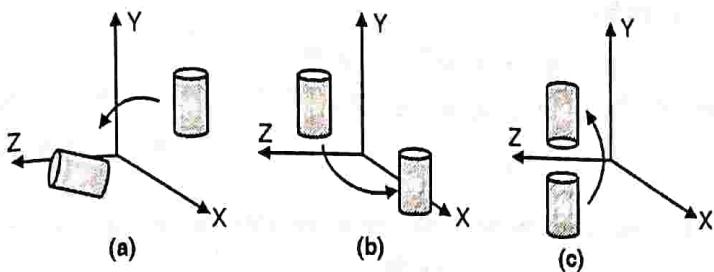
6.4 Rotation

- In 3D, rotation is possible in infinite ways. We can rotate the object around three principal axes, or any arbitrary line.
- Here, we will discuss all possible cases of rotation in space.

Note : Like 2D rotation, for anticlockwise rotation, an angle is considered positive, and for clockwise rotation, the angle is considered negative.

6.4.1 Rotation About Principal Axis

3D rotation is simply an extension of 2D rotation with added dimension. Fig. 6.4.1 depicts the rotation about X, Y and Z-axis.



(a) Rotation about X-axis (b) Rotation about the Y-axis (c) Rotation about Z-axis

Fig. 6.4.1 : 3D Rotation about principal axes

Rotation about X-axis

Rotation of object about X-axis does not alter the X-coordinate. It only affects Y and Z-coordinates. Rotation about X axis is given by,

$$x' = x$$

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

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

Matrix representation of above equation using homogeneous coordinate is given as follow:

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

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

Rotation about Y-axis

Similarly, rotation of an object about Y-axis does not alter the Y-coordinate. It only affects Z and X-coordinates. Rotation about Y-axis is given by,

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

$$y' = y$$

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

Matrix representation of above equation using homogeneous coordinate is given as follow :

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

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

Rotation about Z-axis

- Similarly, rotation of object about Z-axis does not alter the Z-coordinate. It only affects X and Y coordinates.
- Rotation about Z-axis is given by,

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

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

$$z' = z$$

- Matrix representation of above equation using homogeneous coordinate is given as follow :
- Matrix representation of above equation using homogeneous coordinate is given as follow :

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

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

Inverse Rotation Matrix

- Inverse rotation is achieved by negating the rotation angle θ . The transformation matrix for inverse rotation about Z-axis is shown below :

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

- Cosine is even function and sine is an odd function, hence;

$$\cos(-\theta) = \cos \theta \text{ and } \sin(-\theta) = -\sin \theta$$

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

Similarly, we can achieve inverse rotation about X and Y-axis by replacing θ by $-\theta$ in rotation matrix $R_x(\theta)$ and $R_y(\theta)$, respectively.

6.4.2 Rotation About Line Parallel to Principle Axis

- If the rotation axis is not the principal axis, we can make it coincide with one of the principal axes using composite transformation.
- This is done by performing a necessary series of translation and rotation operations.
- If the object is to be rotated about a line parallel to one of the principal axes, we can get the desired rotation with the following steps :
 - o Perform translation such that the rotation axis coincides with one of the principal axes.

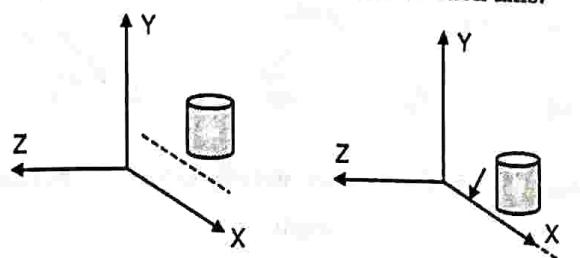
- o Perform specified rotation about that principal axis.
- o Perform inverse translation to move the rotation axis to its original location.
- o Above sequence of steps can be written as :

$$P' = T^{-1} \cdot R_\lambda(\theta) \cdot T \cdot P$$

Where $R_\lambda(\theta)$ indicates the rotation about the principal axis with which rotation axis was aligned. The composite transformation matrix is :

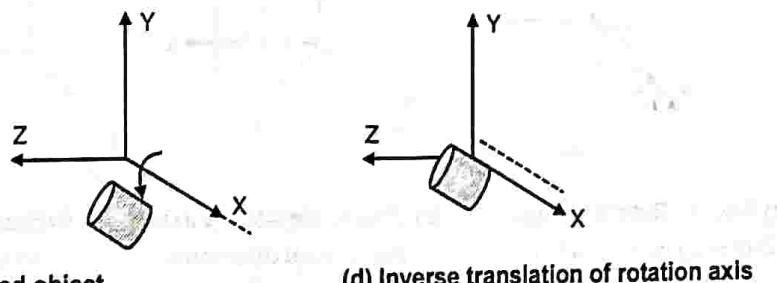
$$M = R(\theta) = T^{-1} \cdot R_\lambda(\theta) \cdot T$$

This operation is the same as 2D rotation about a pivot point in the plane. Fig. 6.4.2 depicts the rotation about a line parallel to X-axis. The dashed line indicates the rotation axis.



(a) Object to be rotated and rotation axis

(b) Translated rotation axis



(c) Rotated object

(d) Inverse translation of rotation axis

Fig. 6.4.2 : Rotation about a line parallel to X-axis

- The same concept can be extended for rotation about a line parallel to Y and Z-axis.

6.4.3 Rotation About an Arbitrary Axis

- In real-world applications, rotation about the principal axis is a rare case. Application programmer needs to rotate the object around any arbitrary axis in space.
- When the object is to be rotated around a line which is not parallel to the principal axis, it requires more efforts.
- This case is more complicated than in previous cases.
- Let the rotation axis passes from the points P_1P_2 as shown in Fig. 6.4.3 (a). Desired rotation is achieved by the composite transformation. Following steps should be performed for that :

Step 1 : Translate the rotation axis such that it passes through the origin (Refer Fig. 6.4.3(b)).

Step 2 : Rotate the translation axis such that it coincides one of the principal axes (Refer Fig. 6.4.3(c)).

Step 3 : Perform the actual rotation operation (Refer Fig. 6.4.3(d)).

Step 4 : Perform inverse rotation to bring rotation axis to the original orientation (Refer Fig. 6.4.3(e)).

Step 5 : Perform inverse translation to move back the rotation axis to its original location (Refer Fig. 6.4.3(f)).

- The sequence of the operations to perform the desired rotation is described in Fig. 6.4.3.

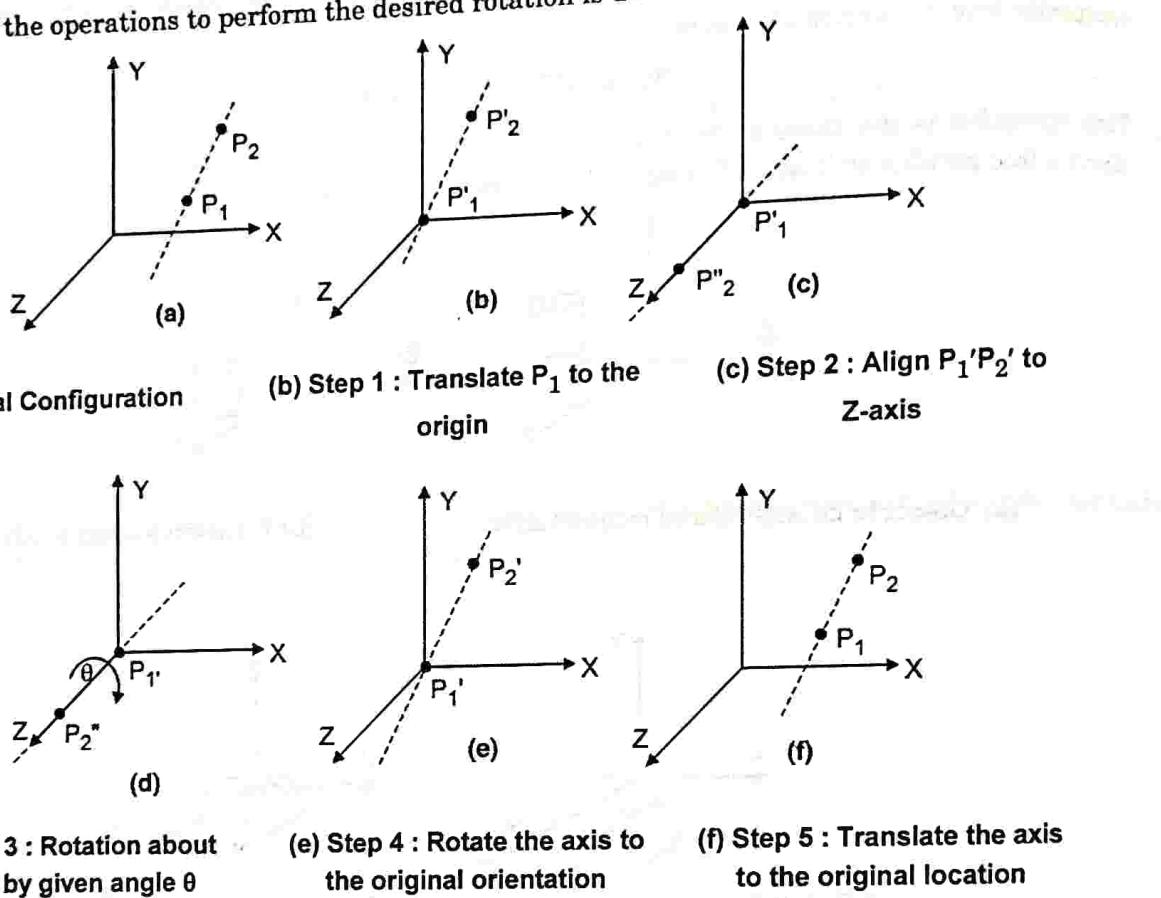


Fig. 6.4.3 : Rotation about an arbitrary line

- We can align the rotation axis with any of the principal axes. For the sake of discussion, we will align it with the Z-axis.

Let $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ be the endpoints of the rotation axis depicted in Fig. 6.4.4.

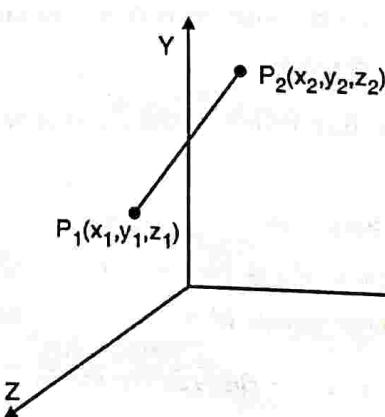


Fig. 6.4.4 : Orientation of rotation axis P_1P_2

Direction of rotation axis P_1P_2 is given by :

$$\overrightarrow{P_1P_2} = P_2 - P_1 = V = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

Unit vector in direction of rotation axis is defined as :

$$u = \frac{V}{|V|} = \frac{x_2 - x_1, y_2 - y_1, z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}} = (a, b, c)$$

$$\text{Where, } a = \frac{x_2 - x_1}{|V|}, b = \frac{y_2 - y_1}{|V|}, c = \frac{z_2 - z_1}{|V|}$$

$$\text{and, } |V| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

If the rotation axis is not passing through the origin, we should translate any of the endpoints (P_1 or P_2) to the origin. Omit this step if rotation axis is already passing through the origin.

For the given case, let us translate the point P_1 to the origin. This is done by applying the following translation matrix to the endpoints of the rotation axis.

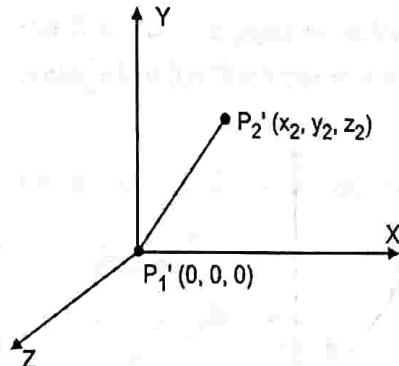
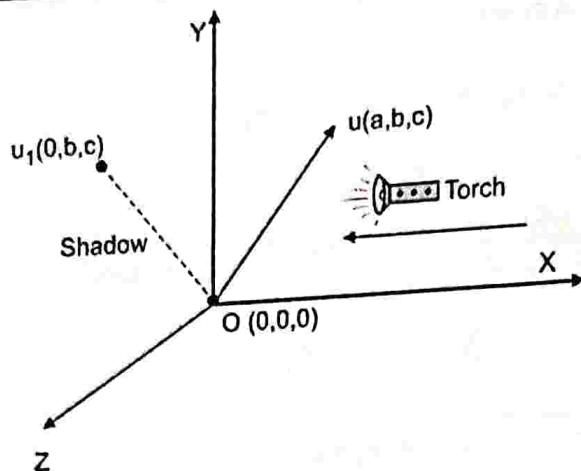


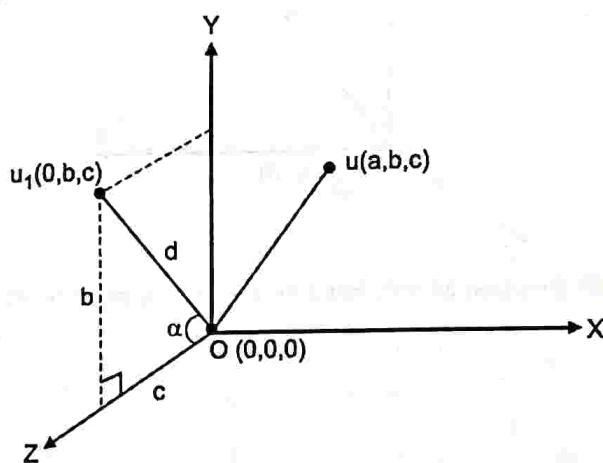
Fig. 6.4.5 : Position of rotation axis after translation by point P_1

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This transformation moves the rotation axis as shown in Fig. 6.4.5, P_1' and P_2' are the translated coordinates of the original endpoints. Length of the rotation axis is already normalized to simplify the mathematical computation. So, normalized coordinates of u would be (a, b, c) as computed before.
- Now align the rotation axis with the Z-axis, to do so rotate the vector u around the X-axis in anticlockwise direction by the angle α such that unit vector falls in XZ plane.
- Then rotate unit vector in XZ plane around the Y-axis in a clockwise direction by the angle β to align it with the Z-axis.
- To align the vector u with Z-axis, first put the torch on X-axis such that the shadow of vector u falls in YZ plane, call the shadow u_1 .
- As this would be parallel projection on the YZ plane, x coordinate will become 0 and the remaining two coordinates will not alter.

Fig. 6.4.6 : Shadow of vector u on YZ plane

- As shown in Fig. 6.4.6, parallel projection of vector u on YZ plane would be $u'(0, b, c)$. Parallel projection on the YZ plane only eliminates the x coordinate; it does not affect remaining coordinates. It is easier to find out an angle α with projected vector u_1 , compared to u .
- As shown in Fig. 6.4.7 Vector u_1 makes an angle α with the Z-axis. If we rotate u and u_1 by the angle α around X-axis, u' will get align with Z-axis and u will fall in XZ plane.

Fig. 6.4.7 : Computation of angle α from projected vector u_1

- From Fig. 6.4.7, sin and cosine angles between u_1 and Z axis is computed using trigonometric rule as,

$$\sin \alpha = \frac{\text{Opposite side}}{\text{Hypotenuse}} = \frac{b}{d}$$

$$\cos \alpha = \frac{\text{Adjacent side}}{\text{Hypotenuse}} = \frac{c}{d}$$

$$d = \sqrt{b^2 + c^2}$$

- The rotation matrix around the X-axis is,

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This transformation matrix moves unit vector u from space to XZ 2D plane as shown in Fig. 6.4.8. Here, u' is the vector after rotation of u around the X-axis by an angle α .

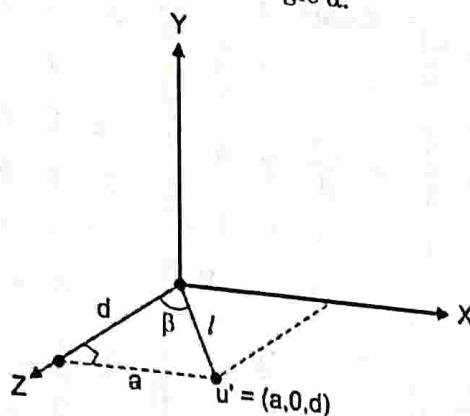


Fig. 6.4.8 : Position of vector u after rotation about X-axis by the angle α

As rotation was performed about X-axis, the x component of u won't change. And u' is in XZ plane, so its y component is zero, and z component is d , we already computed. Rotation does not alter the length and hence the length of vector u' will remain 1.

$$l = \sqrt{a^2 + (b^2 + c^2)} = 1$$

- If we rotate u' by the angle β about Y-axis in a clockwise direction, it will be aligned with the Z-axis.
- From Fig. 6.4.8, sin and cosine angles between u' and Z-axis is computed using the trigonometric rule as,

$$\sin \beta = \frac{\text{Opposite side}}{\text{Hypotenuse}} = \frac{a}{l} = a$$

$$\cos \beta = \frac{\text{Adjacent side}}{\text{Hypotenuse}} = \frac{d}{l} = d$$

- So transformation matrix for rotation about Y-axis would be,

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

Above three steps, translation, rotation about X-axis and rotation about Y-axis would align line P_1P_2 to the Z-axis.

The actual rotation about line P_1P_2 by the angle θ is achieved by rotation about Z-axis using the following rotation matrix.

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

To move the rotation axis to its actual position, perform the first three steps in reverse order. The composite transformation matrix would be,



$$R(\theta) = T^{-1} \cdot R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T$$

$$= \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & \frac{b}{d} & 0 \\ 0 & -\frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Here, } R_y(\beta) \cdot R_x(\alpha) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d & \frac{-ab}{d} & \frac{-ac}{d} & 0 \\ 0 & c/d & \frac{-b}{d} & 0 \\ a & b & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_v$$

$$\text{And } R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & \frac{b}{d} & 0 \\ 0 & -\frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d & 0 & a & 0 \\ \frac{-ab}{d} & \frac{c}{d} & b & 0 \\ \frac{-ac}{d} & \frac{-b}{d} & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_v^T$$

$$\therefore R(\theta) = T^{-1} \cdot A_v^T \cdot R_z(\theta) \cdot A_v \cdot T$$

This is the desired transformation sequence.

Note : If the rotation axis is passing from the origin, then translation and inverse translation steps are not required.

- Assumption : Length of rotation axis is normalized, i.e. $\|\mathbf{v}\| = 1$.

6.5 Reflection

- In 3D, reflection can be performed either with respect to the selected axis or with respect to the selected plane. Reflection about an axis is equivalent to rotation by 180° about that axis.
- Transformation matrices for reflection about $X = 0$, $Y = 0$ and $Z = 0$ planes are shown below :

$$\text{Ref}_{(X=0)} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Ref}_{(Y=0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Ref}_{(Z=0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can compute the reflected coordinate of the object about a plane $X = 0$ by following transformation sequence.

$$P' = M \cdot P = \text{Ref}_{(X=0)} \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$x' = -x$$

$$y' = y$$

$$z' = z$$

- When we take reflection about $X = 0$ plane, only X-coordinate of the object vertices would be flipped, there won't be any change in Y and Z coordinates. We can observe a similar effect for reflection about other planes. Fig. 6.5.1 describes the reflection about $X = 0$ plane. Grey object is the reflected one.

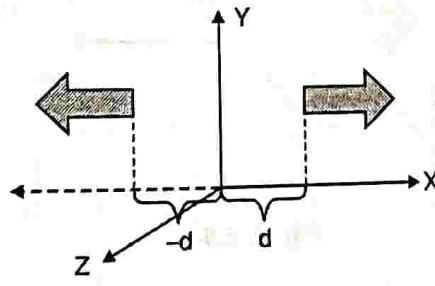


Fig. 6.5.1 : Reflection about $X = 0$ plane

Reflection through the General Plane

Above matrices are valid for reflection through any of the principal planes. We can extend the concept for reflection through arbitrary plane using rotation, translation and reflection.

The plane is uniquely identified by two parameters: the reference point from which the plane is passing and the normal vector of the plane. Let us find the transformation matrix for reflection through a plane passing from reference point R_0 and having normal vector $N = ai + bj + ck$. The transformation is achieved by performing the following steps :

1. Translate the reference point to the origin.



2. Align the normal vector N to Z-axis by performing the following steps.
 - (i) Rotate normal N by the angle α about X-axis such that it falls in XY plane.
 - (ii) Rotate the vector in XY plane about Y-axis by the angle β such that it gets aligned with the Z-axis.
3. Perform reflection through $Z = 0$ plane.
4. Inverse rotation about Y axis by angle β .
5. Inverse rotation about X-axis by angle α .
6. Inverse translation to reposition the reference point to the original location.

So, the transformation matrix would be,

$$M = T^{-1} \cdot R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot \text{Ref}(Z=0) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T$$

If the reference point is specified at the origin, then we do not need to perform step 1 and 6. In that case,

$$M = R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot \text{Ref}(Z=0) \cdot R_y(\beta) \cdot R_x(\alpha)$$

Ex. 6.5.1 : Find the transformation matrix for mirror reflection with respect to the plane passing through the origin and having normal vector in direction $N = i + j + k$.

Soln. :

The geometrical representation of plane passing through the origin and having normal vector in direction $N = i + j + k$ is depicted in Fig. P. 6.5.1.

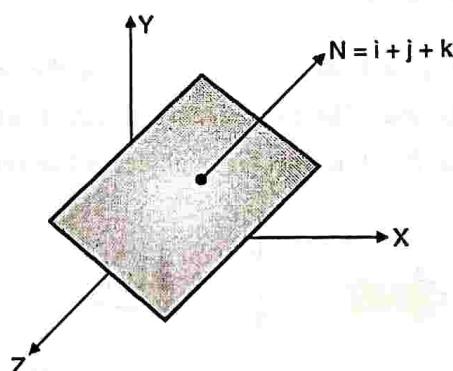


Fig. P. 6.5.1

Perform the following steps to carry out the desired reflection.

1. Rotate normal vector N about X-axis by the angle α so that it falls in XZ plane.

Let us consider the unit normal such that $N = ai + bj + ck$

where $(a, b, c) = (1, 1, 1)$.

$$\cos \alpha = \frac{c}{\sqrt{b^2 + c^2}}, \quad \sin \alpha = \frac{b}{\sqrt{b^2 + c^2}}$$

Let us take $\lambda = d = \sqrt{b^2 + c^2}$ and $|V| = \sqrt{a^2 + b^2 + c^2}$

$$\lambda = d = \sqrt{1+1} = \sqrt{2} \quad \text{and} \quad |V| = \sqrt{1+1+1} = \sqrt{3}$$

2. Rotation about Y-axis in clockwise direction by angle β so that vector gets align with Z-axis.

$$\cos \beta = \frac{\sqrt{b^2 + c^2}}{\sqrt{a^2 + b^2 + c^2}}, \quad \sin \beta = \frac{a}{\sqrt{a^2 + b^2 + c^2}}$$

3. Reflection through XY plane.

4. Inverse rotation about Y-axis by angle β .5. Inverse rotation about X-axis by angle α .

$$M = R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \cdot RF_{(z=0)} \cdot R_y(\beta) \cdot R_x(\alpha) = A_v^T \cdot Ref_{(z=0)} \cdot A_v$$

$$A_v = \begin{bmatrix} \frac{d}{|v|} & \frac{-ab}{d|v|} & \frac{-ac}{d|v|} & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ \frac{a}{|v|} & \frac{b}{|v|} & \frac{c}{|v|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Normal, $N = i + j + k$, so $a = b = c = 1$

$$d = \sqrt{1^2 + 1^2} = \sqrt{2}$$

$$|v| = \sqrt{1^2 + 1^2 + 1^2} = \sqrt{3}$$

$$\therefore A_v = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$RF_{(z=0)} \cdot A_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = A_v^T \cdot Ref_{(z=0)} = A_v = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$M = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} & 0 & 0 \\ -\frac{2}{3} & \frac{1}{3} & 0 & 0 \\ -\frac{2}{3} & -\frac{2}{3} & 1 & 0 \\ \frac{\sqrt{3}}{3} & \frac{2}{3} & 0 & 1 \end{bmatrix}$$

Ex. 6.5.2 : Find the transformation matrix for reflection through the plane passing from reference point $R(x_r, y_r, z_r)$ and having direction vector $N = i + j + k$.

Soln. : The geometrical representation of plane passing through the origin and having normal vector in direction $N = i + j + k$ is depicted in Fig. P. 6.5.2.

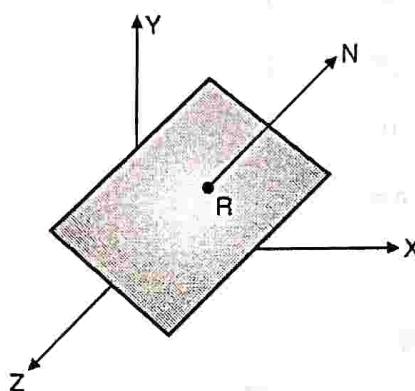


Fig. P. 6.5.2

To perform the desired reflection perform the following steps :

1. Translate reference point to the origin.

$$T = (x_r, y_r, z_r)$$

2. After translation, this case becomes identical to the previous example. To perform steps mentioned in the previous example to derive intermediate transformation matrix.
3. Inverse translation.

$$\begin{aligned} M &= \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} & 0 & 0 \\ -\frac{2}{3} & \frac{1}{3} & 0 & 0 \\ -\frac{2}{3} & -\frac{2}{3} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & 1 & 0 & -y_r \\ 0 & 0 & 1 & -z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} & 0 & -\frac{1}{3}x_r + \frac{2}{3}y_r \\ -\frac{2}{3} & \frac{1}{3} & 0 & -\frac{2}{3}x_r + \frac{1}{3}y_r \\ -\frac{2}{3} & -\frac{2}{3} & 1 & +\frac{2}{3}x_r + \frac{2}{3}y_r - z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} & 0 & \frac{2}{3}(x_r + y_r) \\ -\frac{2}{3} & \frac{1}{3} & 0 & \frac{1}{3}(-2x_r + 4y_r) \\ 0 & 0 & 1 & \frac{2}{3}(x_r + y_r) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.6 Shear

Shearing transformation alters the shape of the object. Shearing is used in 3D viewing to derive projection. Shearing transformation matrix for X, Y and Z axis is shown below :

$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Sh_y = \begin{bmatrix} 1 & c & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$SH_z = \begin{bmatrix} 1 & 0 & e & 0 \\ 0 & 1 & f & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parameters a, b, c, d, e and f can take any real value.

Generalized matrix for 3D shearing is described as,

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

Off diagonal values specify the shear amount in a particular direction. Transformed coordinates after shearing can be computed as,

$$P' = M \cdot P = SH \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & c & e & 1 \\ a & 1 & f & 1 \\ b & d & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\text{Sheared coordinates would be : } P' = [(x + cy + ez) \quad (ax + y + fz) \quad (bx + dy + z)]$$

6.7 Composite Transformations

- Like two dimensional transformation, we can perform complex operations by multiplying sequence of basic transformation matrices in 3D. This is called the **composite transformation**. It is defined as,

$$M = M_n \cdot M_{n-1} \dots M_3 \cdot M_2 \cdot M_1$$

- Each matrix in the sequence represents one basic transformation operation. In column measure representation, i^{th} matrix from the right represents i^{th} transformation operation in order. In short, the composite transformation matrix is not pure operation, but it is composition of various basic operations like translation, rotation, scaling, reflection, shearing etc.

Viewing Pipeline and Coordinates

- In two dimensional graphics, viewing transforms the position of a pixel from world coordinate to device coordinate. Window to viewport transformation maps the world coordinates to device coordinates and performs clipping if necessary. In 3D graphics, more processing is required due to more choices of viewing directions.

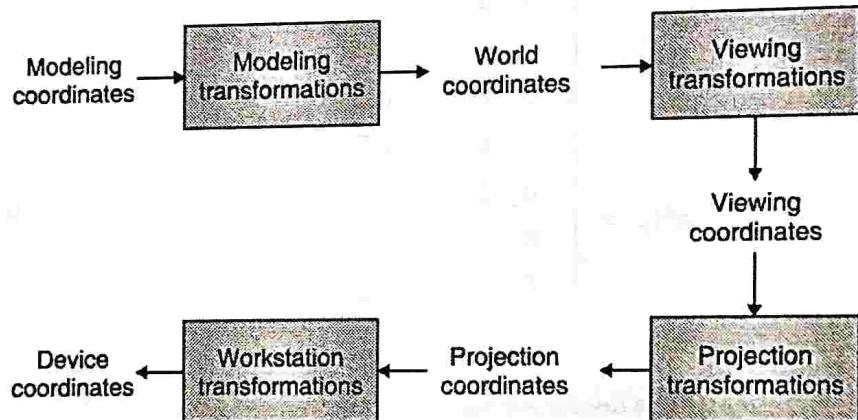
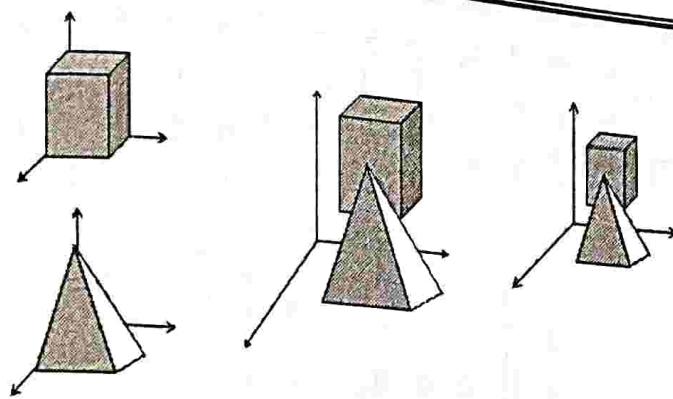


Fig. 6.7.1 : 3D viewing pipeline

- Generation of the scene on the computer is identical to taking photographs. We need to determine the orientation of the camera; we need to decide what portion of world coordinate should be selected etc.
- Fig. 6.7.1 shows the processing steps of the 3D viewing pipeline. Objects in the scene are modelled using modelling coordinates. They define the actual size of the object. Then objects are placed in word using world coordinates. Viewing direction is specified using viewing coordinates. Then the scene is projected on view plane and mapped to device coordinates to display on the screen. Object outside viewing region is clipped. Remaining objects go under the process of visible surface identification and surface rendering.



Modelling coordinates World coordinates Normalized coordinates

Fig. 6.7.2 : Coordinate transformation.

6.8 Solved Examples

Ex 6.8.1: A cube is defined by 8 vertices : A(0, 0, 0), B(2, 0, 0), C(2, 2, 0), D(0, 2, 0), E(0, 0, 2), F(0, 2, 2), G(2, 0, 2), H(2, 2, 2). Perform following 3D transformations on the cube.

(i) Translation by $T = [t_x \ t_y \ t_z] = [5 \ 3 \ 4]$

(ii) Scaling by $S = [s_x \ s_y \ s_z] = [1 \ 2 \ 0.5]$

(iii) Rotation about X-axis by 90° in clockwise direction

Soln. :

Given cube is shown in Fig. P. 6.8.1.

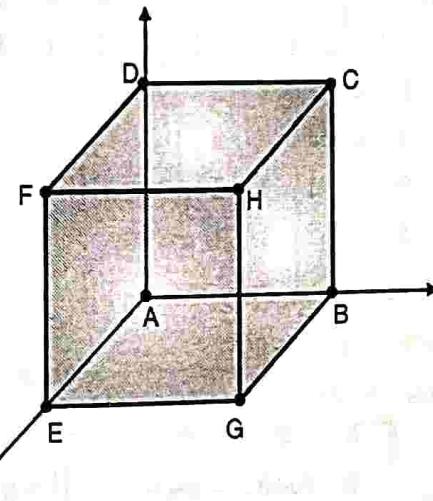


Fig. P. 6.8.1 : $2 \times 2 \times 2$ Cube with A(0, 0, 0) at origin

Let us compute the coordinates for asked operations.

1. Translation by $T = [5 \ 3 \ 4]$

Coordinates of translated cube are computed as,

$$\begin{aligned}
 P' &= T \cdot P = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 P &= \begin{bmatrix} 5 & 7 & 7 & 5 & 5 & 7 & 7 \\ 3 & 3 & 5 & 5 & 3 & 5 & 5 \\ 4 & 4 & 4 & 4 & 6 & 6 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

2. Scaling by $S = [1 \ 2 \ 0.5]$

Coordinates of Scaled cube are computed as,

$$\begin{aligned}
 P' &= S \cdot P = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 P &= \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 4 & 4 & 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

3. Rotation about X-axis by 90° in clockwise direction

Here, rotation is in clockwise direction, so $\theta = -90^\circ$

Coordinates of Scaled cube are computed as,

$$\begin{aligned}
 P' &= R_{x(\theta = -90^\circ)} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90^\circ) & -\sin(-90^\circ) & 0 \\ 0 & \sin\theta(-90^\circ) & \cos(-90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 P = &\begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & -2 & -2 & 0 & -2 & 0 & -2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

Ex 6.8.2: Find a sequence of transformations required to rotate a solid object w.r.t. a line $y = mx + c$ in anticlockwise direction by angle θ .

Soln.: Step 1: Line $y = mx + c$ is shown in Fig. P. 6.8.2. Here, c is the y-intercept translating the line by the amount $[0 \ -c \ 0]$.

We can make the rotation axis pass from the origin by following translation operation.

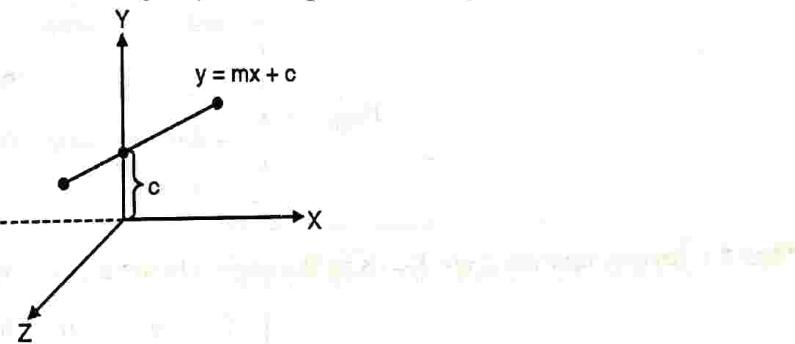


Fig. P. 6.8.2

$$\therefore T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2: Rotate line about X-axis by the angle α in the anticlockwise direction such that it falls in XZ plane.

$$\therefore R_{x(\alpha)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Now rotate line about Y-axis by the angle β in a clockwise direction. Such that it gets aligned with the Z-axis.



$$\therefore R_{y(\beta)} = \begin{bmatrix} \cos(-\beta) & 0 & \sin(-\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\beta) & 0 & \cos(-\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 4 : Now line $y = mx + c$ is aligned with z-axis,

To perform actual rotation about z-axis by angle θ

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

Step 5 : Inverse rotation about Y axis by angle β

$$\therefore R_{y(\beta)}^{-1} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 6 : Inverse rotation about X-axis by the angle α to put a line in its original orientation.

$$\therefore R_{x(\alpha)}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 7 : Inverse translation of line by a vector $[0, c, 0]$ to place it on its actual position.

$$\therefore T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

\therefore Desired transformation sequence to rotate an object about line $y = mx + c$ by angle θ is,

$$M = T^{-1} \cdot R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot R_{z(\theta)} \cdot R_{y(\beta)} \cdot R_{x(\alpha)} \cdot T$$

Ex. 6.8.3 : Rotate a triangle ABC with vertices A(2, 2, 2), B(3, 4, 7) and C(8, 9, 12) about Y-axis in anticlockwise direction by angle 90°.

Soln.:

The rotation angle is 90° and rotation is to be performed in an anticlockwise direction, so $\theta = +90^\circ$. Rotation is to be performed about the Y-axis, so transformed coordinates are computed as,

$$P' = R_y(\theta) \cdot P = \begin{bmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 12 \\ 2 & 4 & 9 \\ -2 & -3 & -8 \end{bmatrix}$$

Rotated vertices are $A'(2, 2, -2)$, $B'(7, 4, -3)$ and $C'(12, 9, -8)$.

Original Coordinates	Transformed Coordinates
$A(2, 2, 2)$	$A'(2, 2, -2)$
$B(3, 4, 7)$	$B'(7, 4, -3)$
$C(8, 9, 12)$	$C'(12, 9, -8)$

Ex 6.8.4 : Find transformation matrix for rotation about a line parallel to the X-axis and passing through the reference point $P(x_r, y_r, z_r)$.

Soln. :

Mentioned composite transformation is achieved by performing the sequence of the following steps :

1. Translation by $T(-x_r, -y_r, -z_r)$
2. Rotation about X-axis by angle θ .
3. Inverse translation.

$$\therefore M = T^{-1} \cdot R_x(\theta) \cdot T = \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & 1 & 0 & -y_r \\ 0 & 0 & 1 & -z_r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & \cos \theta & -\sin \theta & -y_r \cos \theta + z_r \sin \theta \\ 0 & \sin \theta & \cos \theta & -y_r \sin \theta - z_r \cos \theta \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{aligned}
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & -y_r \cos \theta + z_r \sin \theta + y_r \\ 0 & \sin \theta & \cos \theta & -y_r \sin \theta - z_r \cos \theta + z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & z_r \sin \theta + y_r (1 - \cos \theta) \\ 0 & \sin \theta & \cos \theta & z_r (1 - \cos \theta) - y_r \sin \theta \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Ex. 6.8.5 : The pyramid with coordinates A (0, 0, 0), B (1, 0, 0), C (0, 1, 0) and D (0, 0, 1) is to be rotated by about line L that has direction vector $v = j + k$ and passing through point (0, 1, 0). Find the coordinates of the transformed pyramid.

Soln. :

The orientation of said pyramid is shown in Fig. P. 6.8.5.

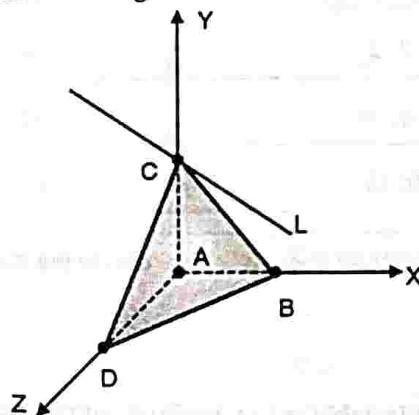


Fig. P. 6.8.5

To rotate the pyramid about line L, we need to align it with one of the principal axes. Let us align it with Z-axis.

We should perform the following steps to achieve the desired operation.

1. Translate line by a vector $(0, -1, 0)$ so that it passes through the origin.

$$\therefore T = [0 \ -1 \ 0]$$

2. Rotate this translated line by an angle α about X-axis in an anticlockwise direction, so that it gets align with Z-axis. Line L is in direction of vector $v = j + k$, so it makes an angle of $\alpha = 45^\circ$ with Y and Z axis and the line is already in YZ plane. So only one rotation is sufficient to align the line with the Z-axis.
3. Perform rotation about Z-axis by angle $\theta = 90^\circ$.
4. Inverse rotation by angle α .
5. Inverse translation by vector T.

$$M = T^{-1} \cdot R_{x(-\alpha)} \cdot R_z(\theta) \cdot R_{x(\alpha)} \cdot T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying first two and last two matrices,

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying last two matrices,

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \\ \frac{-1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformed coordinates are computed as,

$$P' = M \cdot P = \begin{bmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \\ \frac{-1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\therefore P' = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & \frac{2}{\sqrt{2}} \\ \frac{-1}{2} & \frac{\sqrt{2}-1}{2} & 0 & 0 \\ \frac{-1}{2} & -\left(\frac{\sqrt{2}+1}{2}\right) & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A(0, 0, 0)	$A' \left(\frac{1}{\sqrt{2}}, \frac{-1}{2}, \frac{-1}{2} \right)$
B(1, 0, 0)	$B' \left(\frac{1}{\sqrt{2}}, \frac{\sqrt{2}-1}{2}, \frac{\sqrt{2}+1}{2} \right)$
C(0, 1, 0)	$C' (0, 0, 0)$
D(0, 0, 1)	$D' \left(\frac{2}{\sqrt{2}}, 0, 0 \right)$

Ex. 6.8.6 : Find out the 3D transformation matrix to rotate a given 3D object by an amount 60° about a line passing from point $(1, 1, 1)$ and the direction vector $V = 2i + 2j + 2k$.

Soln. :

Here $\theta = 60^\circ$

Line is passing from $(1, 1, 1)$ and its direction vector is $V = 2i + 2j + 2k$. So line interpolates all points having the same x, y and z components hence line also passes through the origin.

$$V = 2i + 2j + 2k = i + j + k = ni + nj + nk$$

$$\text{So, } (a, b, c) = (1, 1, 1)$$

$$\text{As derived in the previous section, } d = \sqrt{b^2 + c^2} = \sqrt{1+1} = \sqrt{2}$$

$$\text{And Length of the vector, } l = |v| = \sqrt{a^2 + b^2 + c^2} = \sqrt{1+1+1} = \sqrt{3}$$

Transformation matrix to rotate an object about the arbitrary line is given as,

$$M = R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot R_{z(\theta)} \cdot R_{y(\beta)} \cdot R_{x(\alpha)} = Av^{-1} \cdot R_z(\theta) \cdot Av$$

Length of vector V is not normalized, so

$$R_{x(\beta)} \cdot R_{x(\alpha)} = A_v = \begin{bmatrix} \frac{d}{|v|} & \frac{-ab}{(d \cdot |v|)} & \frac{-ac}{(d \cdot |v|)} & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ \frac{a}{|v|} & \frac{b}{|v|} & \frac{c}{|v|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} = A_v^T = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{2}\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{\sqrt{2}\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta = 60^\circ) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3} & -\sqrt{3} & 0 & 0 \\ \sqrt{3} & \sqrt{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = A_v^T \cdot R_z(\theta = 60^\circ) \cdot A_v$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{2}\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{\sqrt{2}\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{3} & -\sqrt{3} & 0 & 0 \\ \sqrt{3} & \sqrt{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & \frac{-1}{\sqrt{2}\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying last two matrices,

$$= \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{2}\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{\sqrt{2}\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{2} & \frac{-1-\sqrt{3}}{\sqrt{2}} & \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ \sqrt{2} & \frac{\sqrt{3}-1}{\sqrt{2}} & \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{2\sqrt{3}+1}{3} & -\frac{\sqrt{3}-2}{3} & \frac{4-\sqrt{3}}{3} & 0 \\ \frac{4-\sqrt{3}}{3} & \frac{\sqrt{3}+3}{6} & \frac{7\sqrt{3}-7}{6} & 0 \\ -\frac{\sqrt{3}-2}{3} & \frac{11-5\sqrt{3}}{6} & -\frac{4-2\sqrt{3}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Ex 6.8.7: Rotate a triangle with vertices A(0, 2, 1), B(2, 3, 0) and C(1, 2, 1) by 45° around a line passing through the points (0, 0, 0) and (1, 1, 1).

Soln.:

As derived in previous section, $d = \sqrt{b^2 + c^2} = \sqrt{(0-1)^2 + (0-1)^2} \sqrt{1+1} = \sqrt{2}$

And Length of the rotation axis, $l = |v| = \sqrt{a^2 + b^2 + c^2} = \sqrt{(0-1)^2 + (0-1)^2 + (0-1)^2} = \sqrt{1+1+1} = \sqrt{3}$

The rotation axis is already passing through the origin, so there is no need for translating it. So the desired transformation sequence can be achieved by,

$$M = R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot R_{z(\theta)} \cdot R_{y(\beta)} \cdot R_{x(\alpha)} = Av^{-1} \cdot R_{z(\theta)} \cdot Av$$



$$\text{Where, } \mathbf{Av} = \begin{bmatrix} \frac{d}{|v|} & \frac{-ab}{(d \cdot |v|)} & \frac{-ac}{(d \cdot |v|)} & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ \frac{a}{|v|} & \frac{b}{|v|} & \frac{c}{|v|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{z(\theta = 45^\circ)} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M} = \mathbf{Av}^{-1} \cdot \mathbf{R}_{z(\theta)} \cdot \mathbf{Av}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} 0.815 & 0 & 0.578 & 0 \\ -0.408 & 0.709 & 0.578 & 0 \\ -0.408 & -0.709 & 0.578 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.709 & -0.709 & 0 & 0 \\ 0.709 & 0.709 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.815 & -0.408 & 0.408 & 0 \\ 0 & 0.709 & 0.709 & 0 \\ 0.578 & 0.578 & 0.578 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying last two matrices,

$$\mathbf{M} = \begin{bmatrix} 0.815 & 0 & 0.578 & 0 \\ -0.408 & 0.709 & 0.578 & 0 \\ -0.408 & -0.709 & 0.578 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.578 & -0.792 & 0.213 & 0 \\ 0.578 & 0.213 & -0.792 & 0 \\ 0.578 & 0.578 & 0.578 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} 0.805 & -0.311 & 0.508 & 0 \\ 0.508 & 0.808 & -0.0314 & 0 \\ -0.312 & 0.506 & 0.809 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformed coordinates, $\mathbf{P}' = \mathbf{M} \cdot \mathbf{P}$

$$= \begin{bmatrix} 0.805 & -0.311 & 0.508 & 0 \\ 0.508 & 0.808 & -0.0314 & 0 \\ -0.312 & 0.506 & 0.809 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -0.114 & 0.677 & 0.691 \\ 1.302 & 3.440 & 1.810 \\ 1.821 & 0.894 & 1.509 \\ 1 & 1 & 1 \end{bmatrix}$$

Original Coordinates	Transformed Coordinates
A(0, 2, 1)	A' (-0.114, 1.302, 1.821)
B(2, 3, 0)	B' (0.677, 3.440, 0.894)
C(1, 2, 1)	C' (0.691, 1.810, 1.509)

Ex. 6.8.8: For origin centered unit square, rotate 45° clockwise, scale by a factor 2 in the x-direction. Find resultant coordinates of the square (write required matrices).

Sol.:

The square with size 2 and center at the origin with its vertex coordinates is shown below :

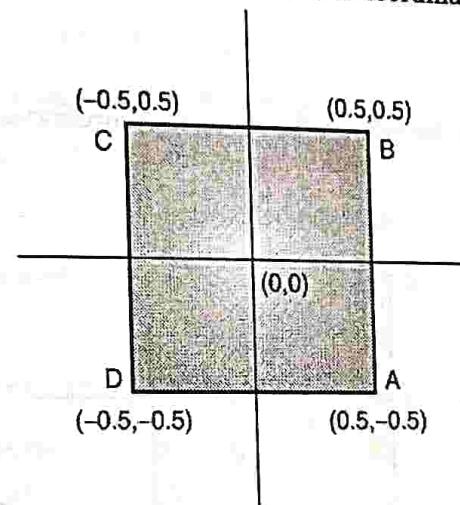


Fig. P. 6.8.8

The square is to be rotated by 45° in the clockwise direction followed by scaled by factor 2 in the x-direction.
 $\therefore \theta = -45^\circ$, $S_x = 2$ and $S_y = 1$.

The coordinates of the rotated square are computed as,

$$\begin{aligned}
 P' &= R \cdot S \cdot P = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 0.71 & 0.71 & 0 \\ -0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 0.71 & 0.71 & 0 \\ -0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 & -1 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$



$$= \begin{bmatrix} 0.355 & 1.065 & -0.355 & -1.065 \\ -1.065 & -0.355 & 1.065 & 0.355 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Thus,

Original Coordinates	Transformed Coordinates
A(0.5, -0.5)	A' (0.355, -1.065)
B(0.5, 0.5)	B' (1.065, -0.355)
C(-0.5, 0.5)	C' (-0.355, 1.065)
D(-0.5, -0.5)	D' (-1.065, 0.355)

Ex. 6.8.9 : Rotate origin centered square with 2 unit length of each side, in the clockwise direction with a rotation angle of 90°.

Soln. :

The square with size 2 and center at the origin with its vertex coordinates is shown below :

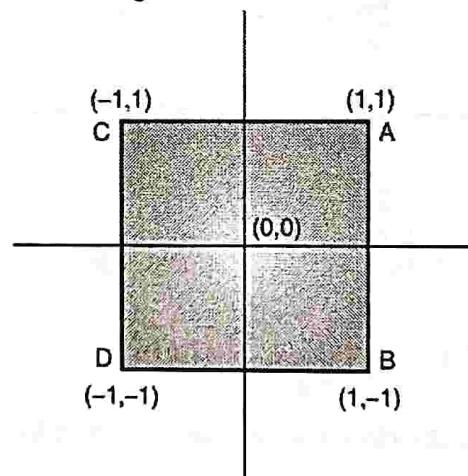


Fig. P. 6.8.9

The square is to be rotated in a clockwise direction, so angle $\theta = -90^\circ$.

The coordinates of a rotated square are computed as,

$$\begin{aligned}
 P' &= M \cdot P = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

Original Coordinates	Transformed Coordinates
A(1, 1)	A' (1, -1)
B(1, -1)	B' (-1, -1)
C(-1, 1)	C' (1, 1)
D(-1, -1)	D' (1, 1)

Review Questions

- Q.1 How to achieve 3D translation?
- Q.2 Derive 3D transformation matrix for translation operation.
- Q.3 How to achieve 3D rotation?
- Q.4 Derive 3D transformation matrix for various cases of 3D rotation.
- Q.5 Derive 3D transformation matrix for rotation about a principal axis.
- Q.6 How to perform 3D rotation about a line parallel to the principal axis?
- Q.7 Derive transformation matrix for 3D rotation about a line parallel to the principal axis.
- Q.8 Derive transformation matrix for rotation about an arbitrary axis in space.
- Q.9 Write matrices in the homogeneous coordinate system for 3D scaling transformation.
- Q.10 How to achieve 3D scaling?
- Q.11 Derive 3D transformation matrices for scaling with respect to the origin and with respect to a reference point.
- Q.12 Derive 3D transformation matrices for scaling with respect to the origin.
- Q.13 Derive 3D transformation matrices for scaling with respect to a reference point.





Module 5

Projection

Syllabus

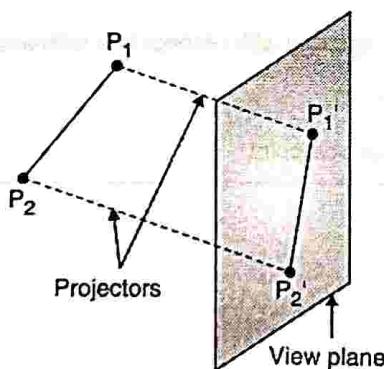
Projections - Parallel, Perspective. (Matrix Representation)

7.1 Introduction

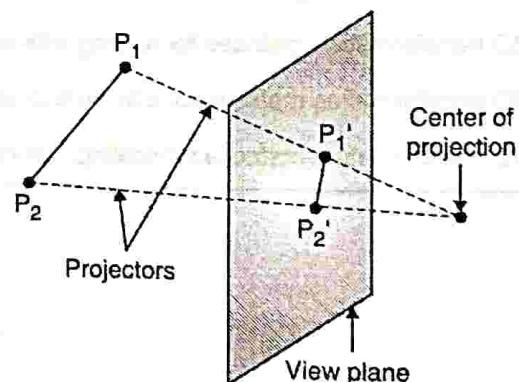
Q. What is meant by parallel and perspective projection?

MU - May 18, Dec. 18, May 19, 10 Marks

- **Projection** is the process of transforming an object representation from n-dimensional space to less than n-dimensional space.
- A 3D object to 2D projection is a vital operation in CAD-CAM and engineering applications.
- **View plane** is the plane on which the object is projected. It is also known as a **projection plane** or plane of projection.
- Projection is broadly classified into two categories : Parallel projection and perspective projection.



(a) Parallel projection



(b) Perspective projection

Fig. 7.1.1 : Types of projection

- In parallel projection, coordinate positions of the object are transformed to view plane by parallel rays/projectors (Refer Fig. 7.1.1 (a)).
- In perspective projection, rays are fired from a point source, called center of projection, which intersects the object coordinates and projects it on view plane (Refer Fig. 7.1.1 (b)).
- Fig. 7.1.1 demonstrates parallel and perspective projections. P_1P_2 is the original line in space. $P'_1P'_2$ is projected line on 2D view plane.
- In the case of parallel projection, projectors are parallel to each other, whereas in case of perspective projection rays are inclined and meeting at a point called **center of projection**.

We can summarize types of projections as follows :

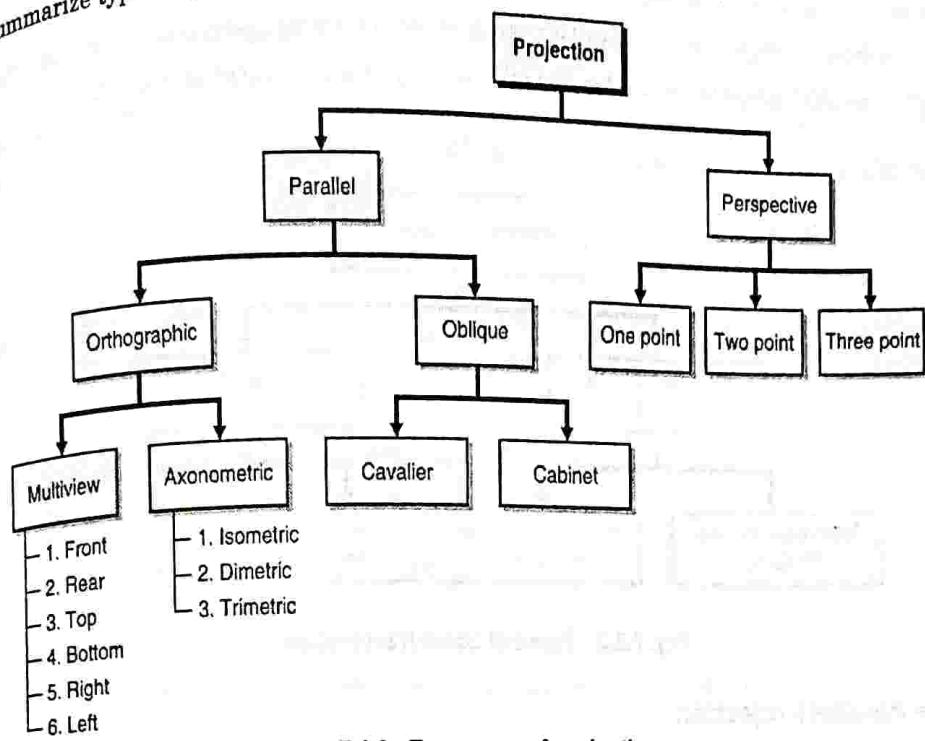
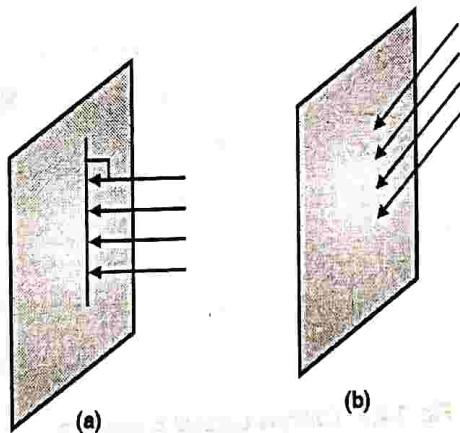


Fig. 7.1.2 : Taxonomy of projection

7.2 Parallel Projection

- Parallel projection is achieved by passing parallel rays from the object vertices and projecting the object on view plane.
- In parallel projection, all projection vectors are parallel to each other.



(a) Orthographic parallel projection

(b) Oblique parallel projection

Fig. 7.2.1 : Types of parallel projection

- Parallel projection preserves true shape and size of the object on view plane.
- However, it fails to capture depth information and hence cannot produce a realistic view.
- Based on the direction of the projection vectors and their relation with view plane, parallel projection is further classified as an orthographic parallel projection or oblique parallel projection.



- When all projectors are parallel to each other and perpendicular to the view plane, it is called **orthographic parallel projection**. If projectors are parallel to each other but are not perpendicular to the view plane, it is called **oblique parallel projection**. Both types of projections are illustrated in Fig. 7.2.1.
- Classification of parallel projection is depicted in Fig. 7.2.2.

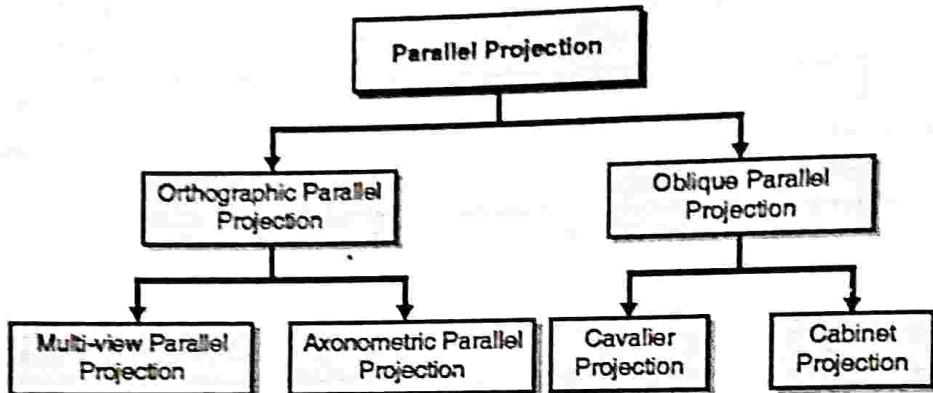


Fig. 7.2.2 : Types of parallel projection

7.2.1 Oblique Parallel Projection

Q. Derive matrix for oblique projection.

MU - May 18, 10 Marks

- As we discussed earlier, in oblique projections, projectors are parallel to each other and they are not perpendicular to the view plane. View in oblique projection is controlled by two parameters ϕ and α as shown in Fig. 7.2.3.

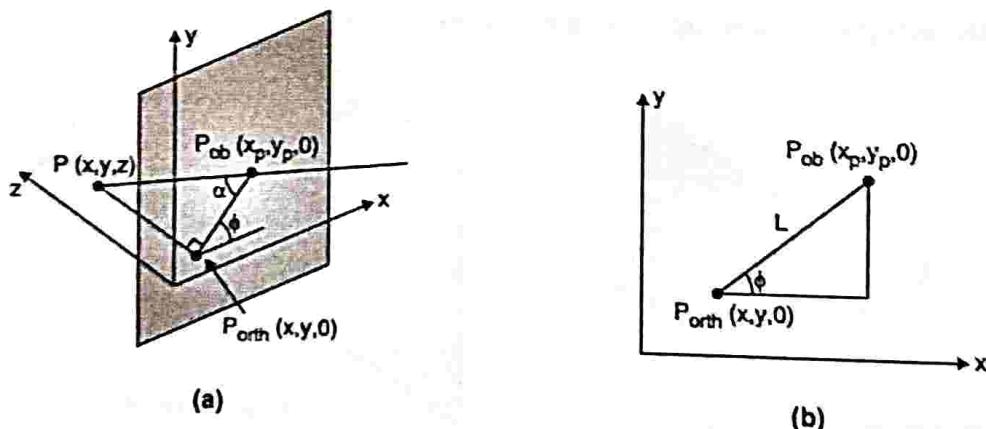


Fig. 7.2.3 : Oblique parallel projection

- In orthographic parallel projection, projectors are perpendicular to the projection plane and hence its corresponding coordinate value becomes zero and the rest of the coordinates will not alter. Whereas in the case of oblique projection, projectors are inclined with the projection plane, which distorts the projected coordinates.
- Let, $P(x, y, z)$ be the point in space. Orthographic projection of point P on view plane XY is $P_{\text{orth}}(x, y, 0)$.
- Oblique projection of P on the same plane is let's say $P_{\text{ob}}(x_p, y_p, 0)$.

Oblique projection vector passing through points P and P_{ob} makes an angle α with view plane. Let the length of line joining points P_{orth} and P_{ob} is L .
 From Fig. 7.2.3(b), values of oblique projection of point $P(x, y, z)$ on XY view plane is given by following equations:

$$x_p = x + L \cos \phi$$

$$y_p = y + L \sin \phi$$

From Fig. 7.2.3 (a),

$$\tan \alpha = \frac{z}{L}$$

$$\therefore L = \frac{z}{\tan \alpha} = zL_1$$

By solving the equations of x_p and y_p for L ,

$$x_p = x + zL_1 \cos \phi$$

$$y_p = y + zL_1 \sin \phi$$

So the transformation matrix for any parallel projection on view plane X_vY_v is written as,

$$M = \begin{bmatrix} 1 & 0 & L_1 \cos \phi & 0 \\ 0 & 1 & L_1 \sin \phi & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L_1 is the foreshortening factor.

Properties

- α is the angle between the oblique projector and plane.
- ϕ is the angle between the horizontal and projected Z-axis.
- $L_1 = 1$ when $\alpha = 90^\circ$, which produces orthographic projection.
- ϕ is a free parameter, but the common choice for ϕ are 30° and 45° .
- Common choice of α are 45° ($\tan \alpha = 1$) and 63.4° ($\tan \alpha = 2$).
- According to the value of α , oblique projection is classified into two categories :
 - o Cavalier projection
 - o Cabinet projection

7.2.1 Cavalier Projection

- If $\alpha = 45^\circ$ (angle between projectors and view plane), the projection is called **cavalier projection**.
- Foreshortening factor for the lines perpendicular to the projection plane would be 1.
- In other words, if the projection angle is 45° , there won't be any change in the projected length of lines which are perpendicular to the view plane.
- Resulting figure in cavalier projection appears thick. 3D nature of the object can be captured but the shape seems to be distorted for lines not parallel to the principal axis (Refer Fig. 7.2.4(a)).



7.2.1.2 Cabinet Projection

- If α is set to 63.4° , the type of projection is known as **cabinet projection**.
- Foreshortening factor for the lines perpendicular to the projection plane would be 0.5.
- In other words, if the projection angle is 63.4° , the projected length of lines perpendicular to the view plane would be half of the original length.
- Cabinet projection is more realistic than cavalier projection.
- Fig. 7.2.4 shows the effect of angle for cabinet and cavalier projection.

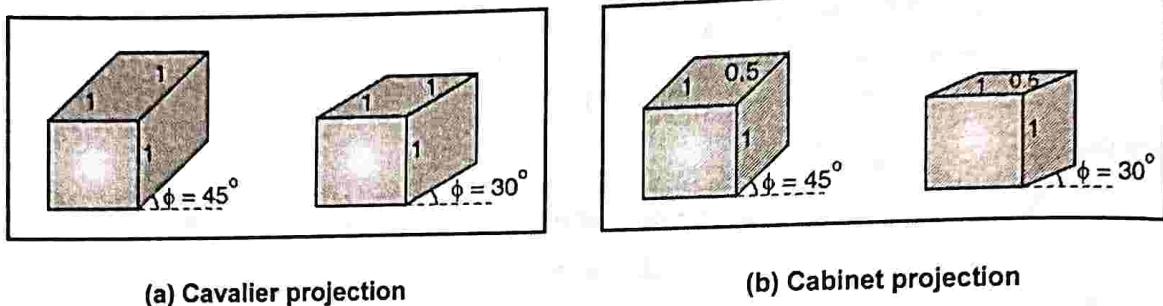


Fig. 7.2.4 : Cabinet and cavalier projection

The relation between angle, projection factor and type of projection is summarized in Table 7.2.1.

Table 7.2.1

The angle between the projector and the view plane(α)	Foreshortening factor	Type of projection
45.0°	1.0	Cavalier
63.4°	0.5	Cabinet
90.0°	0.0	Orthographic

7.2.2 Orthographic Parallel Projection

- Based on the relationship between the principal plane and the sides of the object, we can further classify the orthographic parallel projection in two categories :
 1. Multi-view parallel projection (Sides of the object parallel to principal plane)
 2. Axonometric parallel projection (Sides of the object are not parallel to principal plane)

7.2.2.1 Multiview Parallel Projection

- Multi-view orthographic projection is used in engineering drawings. It is most often used to generate the front view, side view and top view. It is possible to construct the 3D view of an object from these three views.
- In **multi-view parallel projection**, the plane of projection is parallel to the surface of the object. We get only one surface projected on view plane in multi-view projection.
- The front view is called **elevation** and the top view is called a **plan**. Generally, the most descriptive face of an object is selected for elevation. Sometimes, a cross-sectional view is also used to get more details.

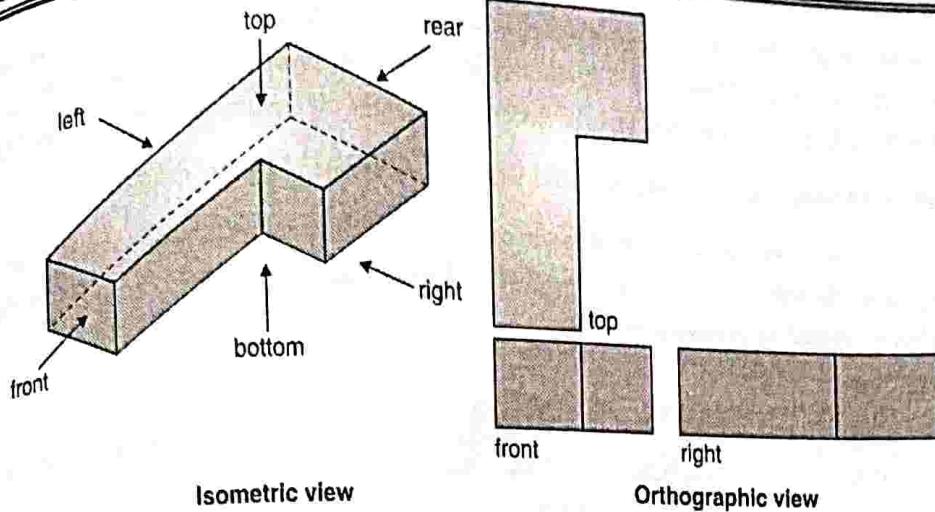


Fig. 7.2.5 : Multi-view parallel projection

On paper, the top view is placed exactly above elevation. Left and right side views are kept on the respective side of the front view.

Orthographic parallel projection preserves true length and the angle between lines, so used to measure the actual dimension of an object.

Fig. 7.2.5 shows the isometric 3D view and its orthographic parallel projection with front, top and right side view.

2.2 Axonometric Parallel Projection

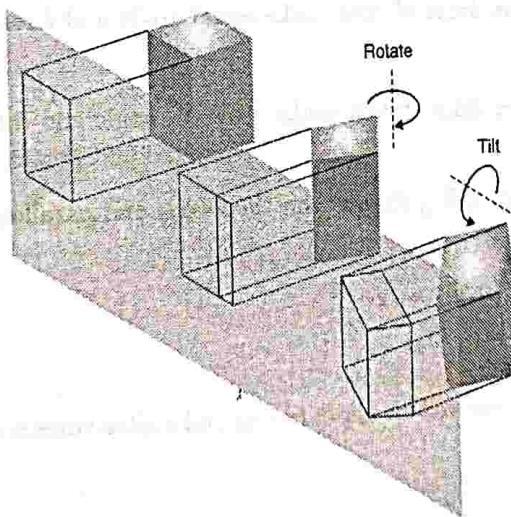


Fig. 7.2.6 : Displaying three surfaces by rotation and tilting

- As shown in Fig. 7.2.6, by performing the rotation of an object, we can display more than one surfaces of an object. Such projection is called **axonometric orthographic projection**.
- Axonometric projection is further classified in isometric, diametric or trimetric by aligning projection plane with three, two or one principal axis respectively.
- Axonometric projection does not preserve the true size of the object. The ratio of projected length to original length is called **foreshortening factor**.



- Isometric projection is achieved by aligning projection vector with the cube diagonal. It preserves the same foreshortening factor for all three axes. For diametric projection, foreshortening factors for any two axes are the same. And for trimetric projection, foreshortening factors for all three principal axes are different.
- Thus, diametric is a special case of trimetric and isometric is a special case of diametric.
- Fig. 7.2.7 explains all three cases of axonometric projection. If diagonal of a unit cube is aligned such that it makes an equal angle with all three axes, projection foreshortens the line with the same factor in all the directions. This is termed as isometric projection.

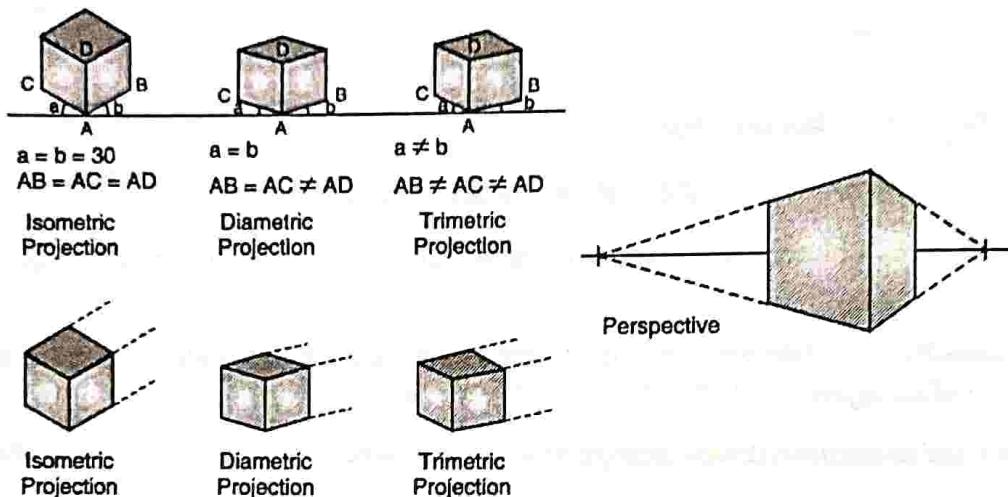


Fig. 7.2.7 : Types of axonometric projection

- In diametric projection, any two sides of cube make equal angle and foreshortening in those two directions would be the same.
- In trimetric projection, all three sides of cube make different angle and their foreshortening factors are also different.
- If the projection plane is placed at Z_{vp} distance on the Z-axis and parallel to XY plane, parallel projection gives the coordinates,

$$x_p = x$$

$$y_p = y$$

- Original z-coordinate value is preserved for depth cueing and visible surface determination.

1. Trimetric Projection

In axonometric projection, at least three faces of an object must be visible. This can be achieved by a first rotating object about Y-axis by angle followed by a rotation about X-axis and then taking projection on $z = 0$ plane.

The transformation matrix for this would be,

$$M = P_z \cdot P_x \cdot P_y(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned}
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \sin \phi & 0 \\ -\sin \theta \cdot \sin \phi & \sin \theta & \cos \theta \cdot \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \sin \phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Projection

Foreshortening ratios are obtained by multiplying a unit vector with the transformation matrix

$$\begin{aligned}
 U = M \cdot I &= \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \cos \phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \cos \phi \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

$$f_x = \sqrt{\cos^2 \theta + \sin^2 \theta \cdot \sin^2 \phi} \quad \dots(7.2.1)$$

$$f_y = \sqrt{\cos^2 \theta} \quad \dots(7.2.2)$$

$$f_z = \sqrt{\sin^2 \theta + \sin^2 \theta \cdot \cos^2 \phi} \quad \dots(7.2.3)$$

Where f_x , f_y and f_z defines foreshortening factors in x, y and z-direction respectively.

For trimetric projection, foreshortening factor for all three directions is different. Trimetric projection is least restrictive.

2. Dimetric Projection

Dimetric projection has more restriction than trimetric projection. In diametric projection, foreshortening factor for any two axes are same, so suppose $f_x = f_y$.

$$\therefore \cos^2 \theta = \cos^2 \phi + \sin^2 \theta \cdot \sin^2 \phi$$

$$\therefore 1 - \sin^2 \theta = 1 - \sin^2 \phi + \sin^2 \theta \cdot \sin^2 \phi$$

$$\sin^2 \phi - \sin^2 \theta \sin^2 \phi = \sin^2 \theta$$

$$\sin^2 \phi (1 - \sin^2 \theta) = \sin^2 \theta$$

$$\sin^2 \phi = \frac{\sin^2 \theta}{1 - \sin^2 \theta} \quad \dots(7.2.4)$$

From Equation (7.2.3),

$$f_z^2 = \sin^2 \phi + \sin^2 \theta \cdot \cos^2 \phi = \sin^2 \phi + \sin^2 \theta \cdot (1 - \sin^2 \phi)$$



$$= \sin^2\phi + \sin^2\theta - \sin^2\theta \cdot \sin^2\phi$$

$$\text{Putting Equation (7.2.4) in this, } f_z^2 = \left(\frac{\sin^2\theta}{1 - \sin^2\theta} \right) + \sin^2\theta - \sin^2\theta \left(\frac{\sin^2\theta}{1 - \sin^2\theta} \right)$$

$$\therefore f_z^2 (1 - \sin^2\theta) = \sin^2\theta + \sin^2\theta (1 - \sin^2\theta) - (\sin^4\theta)$$

$$\therefore f_z^2 - f_z^2 \sin^2\theta = \sin^2\theta + \sin^2\theta - \sin^4\theta - \sin^4\theta$$

$$\therefore 2 \sin^4\theta - \sin^2\theta (f_z^2 + 2) - f_z^2 = 0$$

Solving it for quadratic equation

$$ax^2 + bx + c = 0, a = 2, b = -(f_z^2 + 2), c = f_z^2, x = \sin^2\theta$$

$$\Delta = b^2 - 4ac = (f_z^2 + 2)^2 - 4(2)(f_z^2)$$

$$= f_z^2 + 4f_z^2 + 4 - 8f_z^2 = f_z^2 - 4f_z^2 + 4 = (f_z^2 - 2)^2$$

$$r_1, r_2 = \frac{-b + \sqrt{\Delta}}{2a} = \frac{(-f_z^2 + 2) \pm \sqrt{(f_z^2 - 2)^2}}{4}$$

$$r_1 = \frac{f_z^2 + 2 + f_z^2 - 2}{4} = \frac{f_z^2}{2}$$

$$r_2 = \frac{f_z^2 z^2 + 2 - f_z^2 + 2}{4} = 1$$

But from Equation (7.2.4), $\sin^2\theta = 1$ is not possible

$$\therefore \sin^2\theta = \frac{f_z^2}{2}$$

$$\therefore \theta = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2}} \right)$$

$$\text{From Equation (7.2.4), } \sin^2\phi = \frac{\sin^2\theta}{1 - \sin^2\theta} = \frac{\frac{f_z^2}{2}}{1 - \frac{f_z^2}{2}} = \frac{f_z^2}{2 - f_z^2}$$

$$\phi = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2 - f_z^2}} \right)$$

3. Isometric Projection

Isometric projection is the most restrictive. In isometric projection, foreshortening factor in all three dimensions must be the same.

$$f_x = f_y = f_z$$

From Equation (7.2.2) and (7.2.3),

$$f_y = f_z$$

$$\begin{aligned}
 \cos^2\theta &= \sin^2\phi + \cos^2\phi \cdot \sin^2\theta \\
 1 - \sin^2\theta &= \sin^2\phi + (1 - \sin^2\phi) \cdot \sin^2\theta \\
 1 - \sin^2\theta &= \sin^2\phi + \sin^2\theta - \sin^2\phi \cdot \sin^2\theta \\
 1 - 2 \sin^2\theta &= \sin^2\phi (1 - \sin^2\theta) \\
 \therefore \sin^2\phi &= \frac{1 - 2 \sin^2\theta}{1 - \sin^2\theta}
 \end{aligned} \tag{7.2.5}$$

By comparing R.H.S of Equation (7.2.4) and (7.2.5),

$$\begin{aligned}
 \frac{\sin^2\theta}{1 - \sin^2\theta} &= \frac{1 - 2 \sin^2\theta}{1 - \sin^2\theta} \\
 \therefore \sin^2\theta &= 1 - 2 \sin^2\theta \\
 \therefore \sin^2\theta &= \frac{1}{3} \\
 \therefore \theta &= \sin^{-1}\left(\pm \frac{1}{\sqrt{3}}\right) = \pm 35.26
 \end{aligned}$$

$$\text{From Equation (7.2.4), } \sin^2\phi = \frac{\sin^2\theta}{1 - \sin^2\theta} = \frac{1/3}{1 - 1/3} = \frac{1/3}{2/3} = \frac{1}{2}$$

$$\therefore \phi = \sin^{-1}\left(\pm \frac{1}{2}\right) \pm 45^\circ$$

$$\therefore f_y = f_z$$

$$\therefore f_z = \sqrt{\cos^2\theta} = \sqrt{1 - \sin^2\theta} = \sqrt{1 - \frac{1}{3}} = \frac{\sqrt{2}}{3} = 0.8165$$

Thus isometric is special case of diametric projection with $f_x = f_y = f_z = 0.8165$.

Isometric projection is often used in engineering drawing to construct the 3D view of the object.

7.3 Perspective Projection

Q What is meant by Parallel and Perspective Projections? Derive matrix for Perspective projection.

MU - Dec. 18, May 19, 10 Marks

- Perspective projection preserves the depth information but it does not preserve the true shape and size of the object.
- Size of the projected object depends on the distance between the view plane, the center of projection and the object.
- Projection of a distant object is smaller than that of the nearer object. Fig. 7.3.1 describes this property of perspective projection.

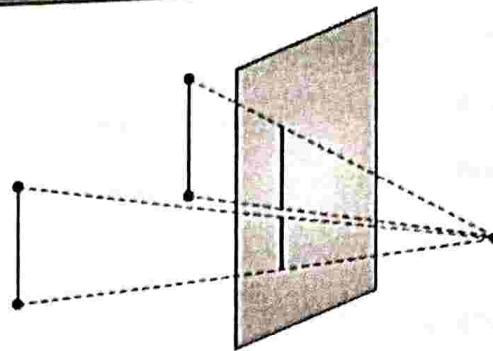


Fig. 7.3.1 : Effect of perspective projection

Let us derive the transformation matrix for the perspective projection of point $P(x, y, z)$ on view plane $Z = 0$ (i.e. XY plane). Assume that the center of projection (COP) is at distance d from the origin on Z-axis as shown in Fig. 7.3.2.

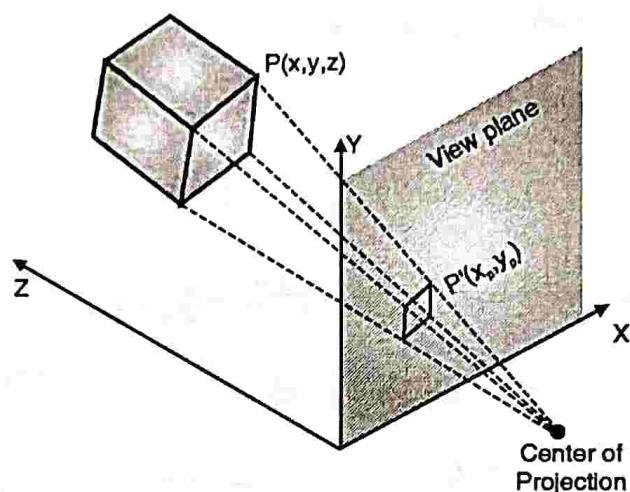
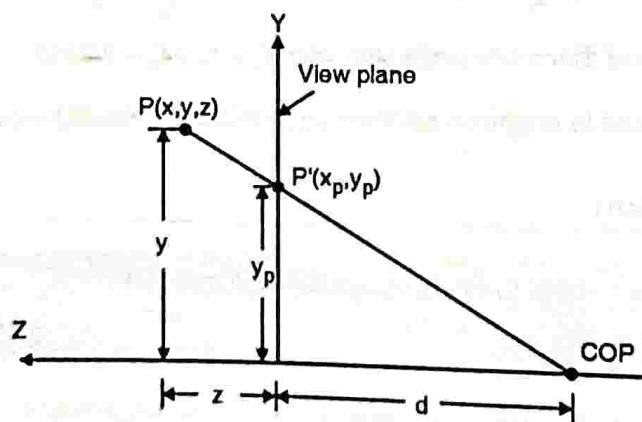
Fig. 7.3.2 : Perspective projection of $P(x, y, z)$ on XY plane

Fig. 7.3.3 : View from X-axis

- For this particular case, we are considering that the object is on positive Z-axis and centre of projection is behind the projection plane i.e. on d distance on negative Z-axis. Although, this can be generalized to any valid position of object, view-plane and COP.
- If we look from the X-axis, Fig. 7.3.2 looks like Fig. 7.3.3.

From the triangle equality rule,

Projection

$$\frac{y_p}{d} = \frac{y}{z+d}$$

$$y_p = \frac{y \cdot d}{z+d}$$

If we look from the Y-axis, Fig. 7.3.2 looks as Fig. 7.3.3.

From the triangle equality rule,

$$\frac{x_p}{d} = \frac{x}{z+d}$$

$$\therefore x_p = \frac{x \cdot d}{z+d}$$

As view plane is positioned at $z = 0$, projected z coordinate would be 0.

$$z_p = 0$$

From this, we can define a transformation matrix as,

$$M = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{bmatrix}$$

Projected Coordinates, $P' = M \cdot P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- If view plane is not XY plane or it is not parallel to XY plane in that case, we need to perform composite transformation such that the normal of the plane get aligned with Z-axis.
- Other cases of perspective projections are discussed at the end of this section.

7.3.1 Vanishing Point

In perspective projection, parallel lines of the object which are not parallel to projection plane are projected into a converging point. In perspective projection, the point where projected parallel lines converge to a single point is called the **vanishing point**.

Projection of railway track forms a vanishing point. It seems they are meeting at a single point at a very far distance. A scene can have any number of vanishing points depending on the orientation of parallel lines in the object.

If the set of parallel lines are parallel to any of the principal axis, vanishing point for such set is called the **principal vanishing point**.

We can control the numbers of principal vanishing points by rotating the view plane. The intersection of the view plane with a number of principal axes defines the number of principal vanishing points.

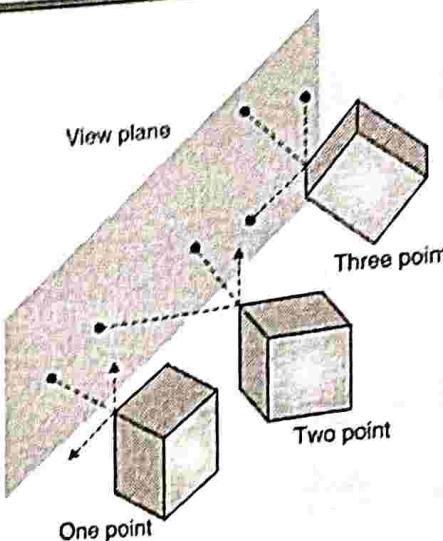


Fig. 7.3.4 : Types of perspective projection

- If view plane intersects to one, two or three principal axes, we have a single point, two points or three-point perspective projection respectively. Fig. 7.3.4 shows the orientation of the cube to achieve one, two and three-point perspective projections.
- Types of perspective projection are given based on a number of the vanishing point. There are mainly three types of perspective projections:
 1. One-point perspective projection
 2. Two-point perspective projection
 3. Three-point perspective projection

7.3.2 Single Point Perspective Projection

- One-point perspective projection is achieved by aligning view plane with two principal axes.
- In other words, the one-point perspective projection is achieved by setting up one of the principal axis perpendiculars to the view plane.
- One-point perspective projection contains only one vanishing point at the horizon. Images like straight roads, railway tracks, buildings etc. have such one-point perspective projection.
- Fig. 7.3.5 explains the one, two and three-point perspective projections.

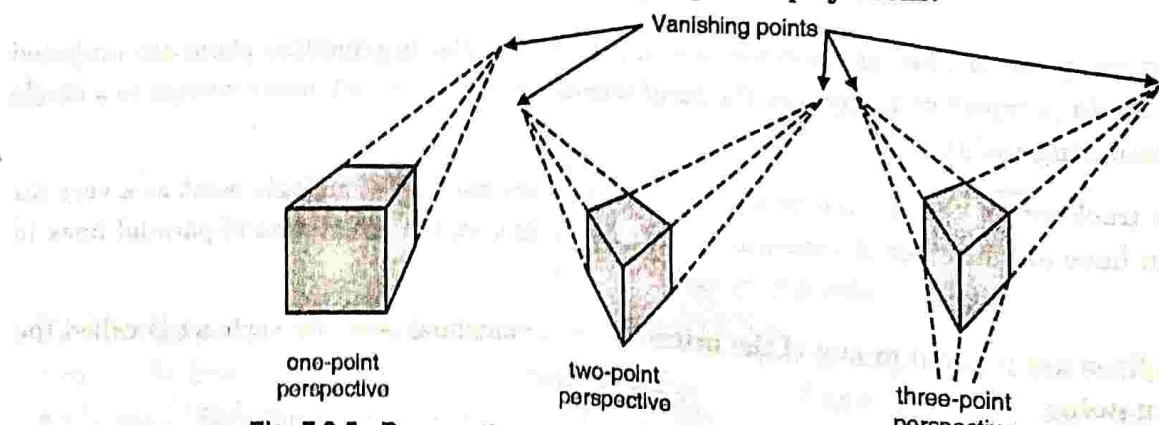


Fig. 7.3.5 : Perspective projection is achieved by projecting the result of the perspective transformation to any of the principal plane

X-direction perspective projection is given by,

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [x \ y \ z \ (px + 1)]$$

$$x' = \frac{x}{px + 1}$$

$$y' = \frac{y}{px + 1}$$

$$z' = \frac{z}{px + 1}$$

Y direction perspective projection is given by,

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [x \ y \ z \ (qy + 1)]$$

$$x' = \frac{x}{qy + 1}$$

$$y' = \frac{y}{qy + 1}$$

$$z' = \frac{z}{qy + 1}$$

Z direction perspective projection is given by,

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [x \ y \ z \ (rz + 1)]$$

$$x' = \frac{x}{rz + 1}$$

$$y' = \frac{y}{rz + 1}$$

$$z' = \frac{z}{rz + 1}$$

7.3.3 Two Point Perspective Projection

Two-point perspective projection occurs when view plane is parallel to one of the principal axes or if view plane intersects exactly two principal axes.

Two-point perspective projection is depicted in Fig. 7.3.5.

When view plan intersects X and Y axis,



X-direction perspective projection is given by,

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [x \ y \ z \ (px + qy + 1)]$$

$$x' = \frac{x}{px + qy + 1}$$

$$y' = \frac{y}{px + qy + 1}$$

$$z' = \frac{z}{px + qy + 1}$$

In a similar way, we can derive the value of (x', y', z') for the other two cases.

7.3.4 Three-Point Perspective Projection

- Three-point perspective projection happens when view plane is not parallel to any of the principal axes
- Three-point perspective projection is depicted in Fig. 7.3.5.
- For Three-point perspective projection,

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [x \ y \ z \ (px + qy + rz + 1)]$$

$$x' = \frac{x}{px + qy + rz + 1}$$

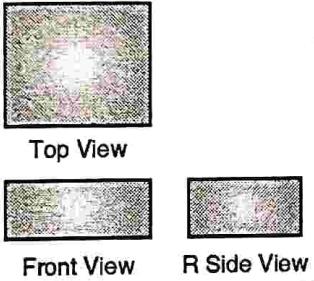
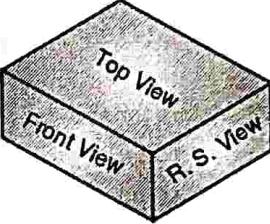
$$y' = \frac{y}{px + qy + rz + 1}$$

$$z' = \frac{z}{px + qy + rz + 1}$$

7.4 Parallel vs. Perspective Projection

Sr. No.	Parallel Projection	Perspective Projection
1.	Projectors are parallel to each other.	Projectors are not parallel.
2.	Need to specify the direction of projection.	Need to specify the center of projection.
3.	Center of projection is at infinite distance.	Center of projection is at a finite distance.
4.	Does not produce a realistic view.	Produces realistic view.
5.	Depth information is lost.	Depth information is preserved.
6.	Preserve relative proportion of object.	Does not Preserve relative proportion of object.
7.	Subtypes : Orthographic projection, oblique projection.	Subtypes : Single point, two points, three-point projection

7.5 Orthographic vs. Isometric Projection

Sr. No.	Orthographic Projection	Isometric Projection
1.	Provides a 2D view of the object	Provides a 3D view of the object
2.	Each view of orthographic projection shows only one side of the object	Isometric projection displays at least three sides of the object
3.	In orthographic projection, the projection plane is parallel to one of the principal plane	In isometric projection, the projection plane is not parallel to any of the principal plane
4.	Does not preserve depth	Does include depth
5.	True shape and size of an object is preserved	The projected object is foreshortened equally in all three directions.
6.	Example :	Example :
	 <p>Top View Front View R Side View</p>	 <p>Top View Front View R. S. View</p>

7.6 Solved Examples

Ex. 7.6.1 : Derive the transformation matrix for parallel projection of point $P(x, y, z)$ on XY plane in direction

$$V = a_i + b_j + c_k$$

Soln. :

$V = a_i + b_j + c_k$ is the direction of projection. Let us assume that parallel projection of point $P(x, y, z)$ on XY plane in direction of V is $P'(x', y', 0)$.

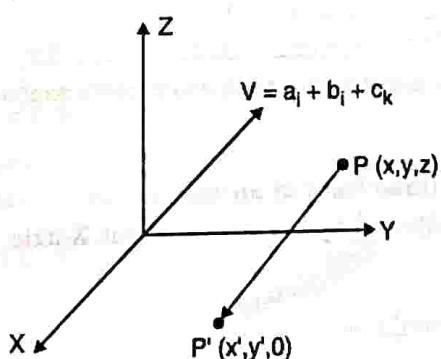


Fig. P. 7.6.1

Projection is in XY plane, so $z' = 0$.

Directions of V and PP' are same. So,



$$\overrightarrow{PP'} = \vec{k} \cdot \vec{V}$$

$$((x' - x), (y' - y), (z' - z)) = k \cdot (a, b, c)$$

$$x' = x + k \cdot a$$

$$y' = y + k \cdot b$$

$$z' = z + k \cdot c$$

$$\text{but, } z' = 0$$

$$\therefore z + k \cdot c = 0$$

$$\therefore k = -\frac{z}{c}$$

$$\therefore x' = x - \frac{a}{c} \cdot z$$

$$\therefore y' = y - \frac{b}{c} \cdot z$$

Transformation matrix for this projection would be,

$$M = \begin{bmatrix} 1 & 0 & -a/c & 0 \\ 0 & 1 & -b/c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix representation of transformed coordinates is given as,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -a/c & 0 \\ 0 & 1 & -b/c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Ex. 7.6.2 : Derive value of rotation angle ϕ and θ for diametric and trimetric axonometric projection.

Soln. :

In axonometric projection, at least three faces of an object must be visible. This can be achieved by a first rotating object about Y-axis by angle followed by a rotation about X-axis and then taking projection on $z = 0$ plane.

The transformation matrix for this would be,

$$M = P_z \cdot P_x \cdot P_y(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned}
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \sin \phi & 0 \\ -\sin \theta \cdot \sin \phi & \sin \theta & \cos \theta \cdot \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \sin \phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Foreshortening ratios are obtained by multiplying a unit vector with the transformation matrix

$$U = M \cdot I$$

$$\begin{aligned}
 &= \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \cos \phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ \sin \theta \cdot \sin \phi & \cos \theta & -\sin \theta \cdot \cos \phi \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

$$f_x = \sqrt{\cos^2 \theta + \sin^2 \theta \cdot \sin^2 \phi} \quad \dots(1)$$

$$f_y = \sqrt{\cos^2 \theta} \quad \dots(2)$$

$$f_z = \sqrt{\sin^2 \theta + \sin^2 \theta \cdot \cos^2 \phi} \quad \dots(3)$$

Where f_x , f_y and f_z defines foreshortening factors in x, y and z-direction respectively.

Foreshortening Factor for Diametric Projection

In diametric projection, foreshortening factor for any two axes are same, so suppose $f_x = f_y$.

$$\therefore \cos^2 \theta = \cos^2 \phi + \sin^2 \theta \cdot \sin^2 \phi$$

$$\therefore 1 - \sin^2 \theta = 1 - \sin^2 \phi + \sin^2 \theta \cdot \sin^2 \phi$$

$$\sin^2 \phi - \sin^2 \theta \sin^2 \phi = \sin^2 \theta$$

$$\sin^2 \phi (1 - \sin^2 \theta) = \sin^2 \theta$$

$$\sin^2 \phi = \frac{\sin^2 \theta}{1 - \sin^2 \theta} \quad \dots(4)$$

From Equation (3),

$$f_z^2 = \sin^2 \phi + \sin^2 \theta \cdot \cos^2 \phi = \sin^2 \phi + \sin^2 \theta \cdot (1 - \sin^2 \phi) = \sin^2 \phi + \sin^2 \theta - \sin^2 \theta \cdot \sin^2 \phi$$



Putting Equation (4) in this,

$$f_z^2 = \left(\frac{\sin^2\theta}{1 - \sin^2\theta} \right) + \sin^2\theta - \sin^2\theta \left(\frac{\sin^2\theta}{1 - \sin^2\theta} \right)$$

$$\therefore f_z^2 (1 - \sin^2\theta) = \sin^2\theta + \sin^2\theta (1 - \sin^2\theta) - (\sin^4\theta)$$

$$\therefore f_z^2 - f_z^2 \sin^2\theta = \sin^2\theta + \sin^2\theta - \sin^4\theta - \sin^4\theta$$

$$\therefore 2 \sin^4\theta - \sin^2\theta (f_z^2 + 2) - f_z^2 = 0$$

Solving it for quadratic equation

$$ax^2 + bx + c = 0, a = 2, b = -(f_z^2 + 2), c = f_z^2, x = \sin^2\theta$$

$$\Delta = b^2 - 4ac = (f_z^2 + 2)^2 - 4(2)(f_z^2)$$

$$= f_z^2 + 4f_z^2 + 4 - 8f_z^2 = f_z^2 - 4f_z^2 + 4 = (f_z^2 - 2)^2$$

$$r_1, r_2 = \frac{-b + \sqrt{\Delta}}{2a} = \frac{(-f_z^2 + 2) \pm \sqrt{(f_z^2 - 2)^2}}{4}$$

$$r_1 = \frac{f_z^2 + 2 + f_z^2 - 2}{4} = \frac{f_z^2}{2}$$

$$r_2 = \frac{f_z^2 z^2 + 2 - f_z^2 + 2}{4} = 1$$

But from Equation (4), $\sin^2\theta = 1$ is not possible

$$\therefore \sin^2\theta = \frac{f_z^2}{2}$$

$$\therefore \theta = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2}} \right)$$

$$\text{From Equation (4), } \sin^2\phi = \frac{\sin^2\theta}{1 - \sin^2\theta} = \frac{\frac{f_z^2}{2}}{1 - \frac{f_z^2}{2}} = \frac{f_z^2}{2 - f_z^2}$$

$$\phi = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2 - f_z^2}} \right)$$

Foreshortening Factor for Isometric Projection

In isometric projection, foreshortening factor in all three dimensions must be same.

$$f_x = f_y = f_z$$

From Equation (2) and (3),

$$\begin{aligned}
 f_y &= f_z \\
 \cos^2\theta &= \sin^2\phi + \cos^2\phi \cdot \sin^2\theta \\
 1 - \sin^2\theta &= \sin^2\phi + (1 - \sin^2\phi) \cdot \sin^2\theta \\
 1 - \sin^2\theta &= \sin^2\phi + \sin^2\theta - \sin^2\phi \cdot \sin^2\theta \\
 1 - 2 \sin^2\theta &= \sin^2\phi (1 - \sin^2\theta) \\
 \therefore \sin^2\phi &= \frac{1 - 2 \sin^2\theta}{1 - \sin^2\theta} \quad \text{Projection} \\
 \end{aligned} \tag{5}$$

By comparing R.H.S of Equation (4) and (5),

$$\begin{aligned}
 \frac{\sin^2\theta}{1 - \sin^2\theta} &= \frac{1 - 2 \sin^2\theta}{1 - \sin^2\theta} \\
 \therefore \sin^2\theta &= 1 - 2 \sin^2\theta \\
 \therefore \sin^2\theta &= \frac{1}{3} \\
 \therefore \theta &= \sin^{-1}\left(\pm \frac{1}{\sqrt{3}}\right) = \pm 35.26
 \end{aligned}$$

From Equation (4),

$$\sin^2\phi = \frac{\sin^2\theta}{1 - \sin^2\theta} = \frac{1/3}{1 - 1/3} = \frac{1/3}{2/3} = \frac{1}{2}$$

$$\therefore \phi = \sin^{-1}\left(\pm \frac{1}{2}\right) \pm 45^\circ$$

$$\therefore f_y = f_z$$

$$\therefore f_z = \sqrt{\cos^2\theta} = \sqrt{1 - \sin^2\theta} = \sqrt{1 - \frac{1}{3}} = \frac{\sqrt{2}}{3} = 0.8165$$

Thus isometric is special case of diametric projection with $f_x = f_y = f_z = 0.8165$.

For trimetric projection, foreshortening factor for all three directions is different. Trimetric projection is less restrictive.

Ex. 7.6.3 : Find transformed co-ordinate of a unit cube with one vertex at origin for,

- (i) Cavalier projection with $\phi = 45^\circ$ (ii) Cabinet projection with $\phi = 30^\circ$

Soln.: Let us assume that vertex A of unit cube is positional at the origin.

From Fig. P. 7.6.3,

- A = (0, 0, 0)
- B = (1, 0, 0)
- C = (1, 1, 0)
- D = (0, 1, 0)
- E = (0, 0, 1)
- F = (1, 0, 1)
- G = (1, 1, 1)
- H = (0, 1, 1)

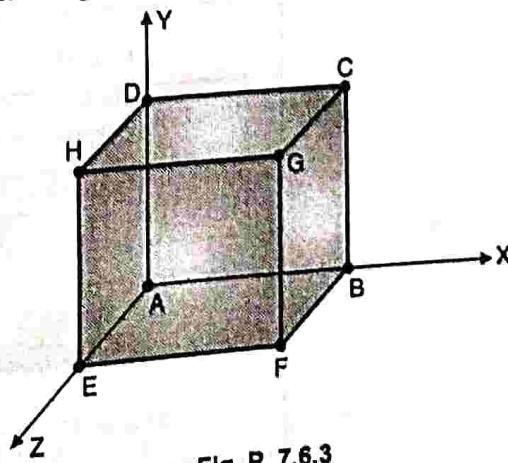


Fig. P. 7.6.3



(i) Cavalier projection

For cavalier projection, foreshortening factor $L_1 = 1$

Transformation matrix for cavalier and cabinet projection is,

$$M = \begin{bmatrix} 1 & 0 & L_1 \cos \phi & 0 \\ 0 & 1 & L_1 \sin \phi & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With $\phi = 45^\circ$ and $L_1 = 1$

$$M = \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformed points are computed as,

$$P' = M \cdot P$$

$$M = \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

By solving above matrices,

Original Coordinates	Transformed Coordinates
A(0, 0, 0)	A' (0, 0, 0)
B(1, 0, 0)	B' (1, 0, 0)
C(1, 1, 0)	C' (1, 1, 0)
D(0, 1, 0)	D' (0, 1, 0)
E(0, 0, 1)	E' $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$
F(1, 0, 1)	F' $\left(\frac{\sqrt{2}+1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$

Original Coordinates	Transformed Coordinates
G(1, 1, 1)	$G' \left(\frac{\sqrt{2}+1}{\sqrt{2}}, \frac{\sqrt{2}+1}{\sqrt{2}}, 0 \right)$
H(0, 1, 1)	$H' \left(\frac{1}{\sqrt{2}}, \frac{\sqrt{2}+1}{\sqrt{2}}, 0 \right)$

(ii) Cabinet projection

For cabinet projection, foreshortening factor, $f = L_1 = \frac{1}{2}$

Given angle $\phi = 30^\circ$

$$\text{So, } M = \begin{bmatrix} 1 & 0 & L_1 \cos \phi & 0 \\ 0 & 1 & L_1 \sin \phi & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{\sqrt{3}}{4} & 0 \\ 0 & 1 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformed coordinates of the cube are computed as,

$$\begin{aligned} P' &= M \cdot P \\ &= \begin{bmatrix} 1 & 0 & \frac{\sqrt{3}}{4} & 0 \\ 0 & 1 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

- By solving the above matrices,

Original Coordinates	Transformed Coordinates
A (0, 0, 0)	A' (0, 0, 0)
B (1, 0, 0)	B' (1, 0, 0)
C (1, 1, 0)	C' (1, 1, 0)
D (0, 1, 0)	D' (0, 1, 0)
E (0, 0, 1)	$E' \left(\frac{\sqrt{3}}{4}, \frac{1}{4}, 0 \right)$
F (1, 0, 1)	$F' \left(\frac{4+\sqrt{3}}{4}, \frac{1}{4}, 0 \right)$



Original Coordinates	Transformed Coordinates
G (1, 1, 1)	$G' \left(\frac{4 + \sqrt{3}}{4}, \frac{5}{4}, 0 \right)$
H (0, 1, 1)	$H' \left(\frac{\sqrt{3}}{4}, \frac{5}{4}, 0 \right)$

Ex. 7.6.4 : Derive a transformation matrix for perspective projection when the center of projection is at origin and plane of projection is at distance d on the Z-axis.

Soln. :

Spatial orientation of the viewplane is shown in Fig. P. 7.6.4.

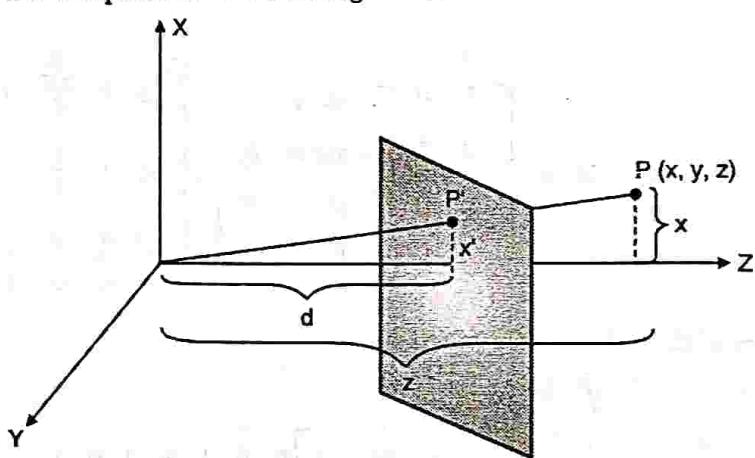


Fig. P. 7.6.4

Suppose projection of point $P(x, y, z)$ on given view plane is $P'(x', y', z')$.

From the rule of triangle equality,

$$\frac{x}{z} = \frac{x'}{d}$$

$$\therefore x' = \frac{x}{z} \cdot d$$

$$\text{Similarly, } \frac{y}{z} = \frac{y'}{d}$$

$$\therefore y' = \frac{y}{z} \cdot d$$

And z' would be d because view plane is on Z-axis at distance d .

$$\therefore z' = d$$

Transformation matrix would be,

$$M = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P' = M \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

P. 7.6.5: Using origin as the center of projection, derive transformation matrix for perspective projection on a view plane passing through reference point $R_0(x_0, y_0, z_0)$ and having normal vector $\bar{N} = n_1 i + n_2 j + n_3 k$.

Soln:

Spatial orientation of the viewplane is shown in Fig. P. 7.6.5.

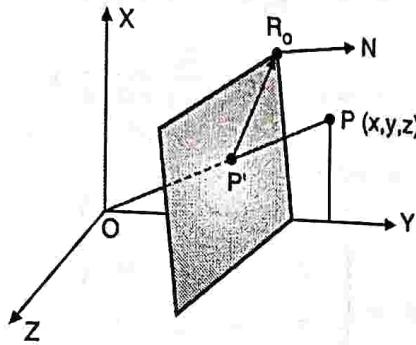


Fig. P. 7.6.5

Let us assume that projection of point $P(x, y, z)$ on the given view plane is $P'(x', y', z')$.

From Fig. P. 7.6.5, OP' and OP both vectors have the same direction but different magnitude.

$$\therefore \overrightarrow{OP'} = \alpha \overrightarrow{OP}$$

$$(P' - O) = \alpha (P - O)$$

$$\therefore (x', y', z') = \alpha (x, y, z)$$

$$x' = \alpha x; y' = \alpha y; z' = \alpha z$$

We need to find out α to find projected co-ordinates.

R_0 and P' are on the view plane and \bar{N} is normal to the plane. The dot product of two perpendicular vectors is always zero. So,

$$(\overrightarrow{R_0 P'}) \cdot \bar{N} = 0$$

$$(P' - R_0) \cdot \bar{N} = 0$$

$$(x', y', z') - (x_0, y_0, z_0) \cdot (n_1, n_2, n_3) = 0$$

$$(x' - x_0) n_1 + (y' - y_0) n_2 + (z' - z_0) n_3 = 0$$

$$n_1 x' + n_2 y' + n_3 z' = n_1 x_0 + n_2 y_0 + n_3 z_0$$

Put, $x' = \alpha x, y' = \alpha y$ and $z' = \alpha z$ in above equation.



$$n_1\alpha x + n_2\alpha y + n_3\alpha z = n_1 x_0 + n_2 y_0 + n_3 z_0$$

$$\alpha(n_1x + n_2y + n_3z) = n_1 x_0 + n_2 y_0 + n_3 z_0$$

$$\alpha = \frac{n_1 x_0 + n_2 y_0 + n_3 z_0}{n_1 x + n_2 y + n_3 z}$$

$$\text{Let, } d_0 = n_1 \cdot x_0 + n_2 \cdot y_0 + n_3 \cdot z_0$$

$$\alpha = \frac{d_0}{n_1 x + n_2 y + n_3 z}$$

$$x' = \alpha x = \frac{d_0}{n_1 x + n_2 y + n_3 z} \cdot x$$

$$y' = \alpha y = \frac{d_0}{n_1 x + n_2 y + n_3 z} \cdot y$$

$$z' = \alpha z = \frac{d_0}{n_1 x + n_2 y + n_3 z} \cdot z$$

Transformation matrix for $P' = M \cdot P$ would be,

$$M = \begin{bmatrix} d_0 & 0 & 0 & 0 \\ 0 & d_0 & 0 & 0 \\ 0 & 0 & d_0 & 0 \\ n_1 & n_2 & n_3 & 0 \end{bmatrix}$$

- Ex. 7.6.6 :** Derive a general perspective transformation matrix for projection on a plane passing from reference point $R_0(x_0, y_0, z_0)$ having a normal vector $N = n_1 i + n_2 j + n_3 k$. Consider centre of projection at $C(a, b, c)$.

Soln. :

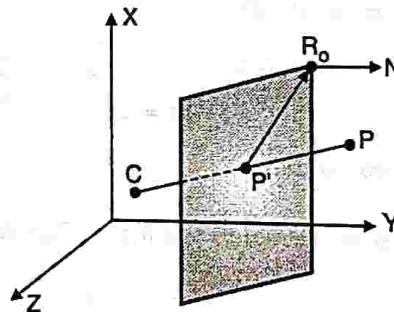


Fig. P. 7.6.6

Spatial orientation of the view plane is shown in Fig. P. 7.6.6.

Coordinate of center of projection is $C(a, b, c)$. Let the point P be (x, y, z) and its projection on view plane is $P'(x', y', z')$.

From Fig. P. 7.6.6,

$$\overrightarrow{CP'} = \alpha \overrightarrow{CP}$$

$$\begin{aligned}
 (P' - C) &= \alpha (P - C) \\
 (x', y', z') - (a, b, c) &= \alpha [(x, y, z) - (a, b, c)] \\
 (x' - a, y' - b, z' - c) &= \alpha (x - a, y - b, z - c) \\
 x' &= a + \alpha (x - a) \\
 y' &= b + \alpha (y - b) \\
 z' &= c + \alpha (z - c)
 \end{aligned}$$

Dot product of two perpendicular vectors is zero.

$$\therefore (\overrightarrow{R_0 P'}) \cdot N = 0$$

$$(P' - R_0) \cdot N = 0$$

$$\{(x', y', z') - (x_0, y_0, z_0)\} \cdot (n_1, n_2, n_3) = 0$$

$$(x' - x_0) n_1 + (y' - y_0) n_2 + (z' - z_0) n_3 = 0$$

$$x' n_1 + y' n_2 + z' n_3 = n_1 x_0 + n_2 y_0 + n_3 z_0$$

Replacing value of x' , y' and z'

$$(a + \alpha (x - a)) n_1 + (b + \alpha (y - b)) n_2 + (c + \alpha (z - c)) = n_1 x_0 + n_2 y_0 + n_3 z_0$$

Rearranging the terms,

$$\alpha (n_1 (x - a) + n_2 (y - b) + n_3 (z - c)) = (n_1 x_0 + n_2 y_0 + n_3 z_0) - (a \cdot n_1 + b \cdot n_2 + c \cdot n_3)$$

Let us take $d_0 = n_1 x_0 + n_2 y_0 + n_3 z_0$ and,

$$d_1 = n_1 a + n_2 b + n_3 c$$

$$\alpha = \frac{d_0 - d_1}{(x - a) \cdot n_1 + (y - b) \cdot n_2 + (z - c) \cdot n_3}$$

And from previous comparison,

$$x' = (x - a) \cdot \alpha + a$$

$$y' = (y - b) \cdot \alpha + b$$

$$z' = (z - c) \cdot \alpha + c$$

From this, transformation matrix is written as,

$$M = \begin{bmatrix} d_0 + a n_1 & a n_2 & a n_3 & (d_0 - a) + a ((n_1 - a) + (n_2 - b) + (n_3 - c)) \\ b n_1 & d_0 + b n_2 & b n_3 & (d_0 - b) + b ((n_1 - a) + (n_2 - b) + (n_3 - c)) \\ c n_1 & c n_2 & d_0 + c n_3 & (d_0 - c) + c ((n_1 - a) + (n_2 - b) + (n_3 - c)) \\ n_1 & n_2 & n_3 & (n_1 - a) + (n_2 - b) + (n_3 - c) \end{bmatrix}$$

Second Method

1. Translate C to the origin.
2. This is identical to the previous case. Use that transformation matrix.
3. Inverse translation of C.



$$M = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 & 0 & 0 & 0 \\ 0 & d_0 & 0 & 0 \\ 0 & 0 & d_0 & c \\ n_1 & n_2 & n_3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Review Questions

- Q. 1** What is the projection? Why projection is necessary?
- Q. 2** Define the terms : Projection, view plane, Direction of projection, the center of projection.
- Q. 3** Enlist the types of projection.
- Q. 4** Write a short note on parallel projection.
- Q. 5** Explain orthographic parallel protection.
- Q. 6** Explain in detail Oblique projection.
- Q. 7** Write a short note on cavalier projection.
- Q. 8** Write a short note on cabinet projection.
- Q. 9** Write a short note on perspective projection.
- Q. 10** Derive a perspective projection of point P (x, y, z) on a view plane positioned at z = 0 and center of projection is on negative z-axis at distance d.
- Q. 11** Write a short note on one-point perspective projection.
- Q. 12** Write a short note on Two-point perspective projection.
- Q. 13** Write a short note on Three-point perspective projection
- Q. 14** Differentiate : Parallel projection vs. Perspective projection.
- Q. 15** Explain types of Parallel Projection with example.



Curves and Fractals

Module 5

Syllabus

Bezier Curve, B-Spline Curve, Fractal-Geometry, Fractal Dimension, Koch Curve

8.1 Curves

8.1.1 Introduction

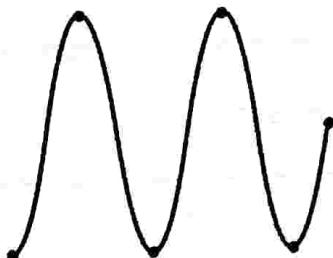
- Real-world objects are not always made up of basic primitives like line, circle, ellipse, square etc.
- Usually, natural objects are very irregular in shape and size. It may not be possible to describe such shapes with standard mathematical formulas.
- Through the curve generation procedure, it is possible to model any random shape. However, the process of curve generation requires detail mathematical analysis of object shape. The process is very complex.
- 3D lines, curves or surfaces are produced either using the mathematical function or by defining a set of data points, called control points. Curves or surfaces may be represented using the parametric or non-parametric form.
- When the function is used to plot the curve, graphics packages evaluate predefined equations for passed arguments and generate pixels along the desired path. The surface is often derived with the help of tessellation. It is first approximated as a polygon mesh. This is done by triangulation. Each triangle is defined using three vertices. Four or more vertices may not form a single plane. Quadratic and super quadratics are produced using this approach.
- When an object is defined using a set of discrete points, the shape is estimated by regression methods. Regression techniques try to generate a best-fitting curve for a given set of control points. This is useful in digitizing coordinates, to design new shapes, to define a smooth path for animation motion etc.

8.1.2 Interpolation and Approximation

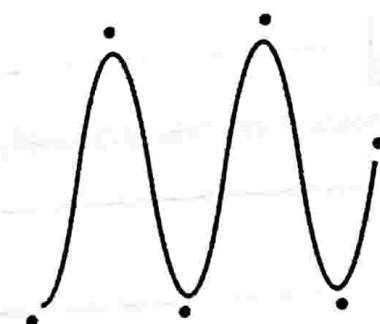
- As we discussed, the curve is specified by a set of control points. Position of control points controls the shape of the curve. Fitting a single curve of the higher degree to all the control points is difficult. Normally, the big curve is approximated by joining piecewise polynomials of lower degree, which approximates few control points only.
- If the curve passes through the control points, it is called interpolation. Interpolation curves are used in animation and specifying camera motion. It is also used in digitizing the coordinates.



- If the curve does not pass through the control points and approximate the shape, it is called **approximation** or **extrapolation**. Such curves are used to estimate the shape of the object surface. Fig. 8.1.1 shows the geometry of both methods.
- Manipulating spline curves using control point representation is very easy. The designer can adjust the shape of the curve by repositioning the control point. It is highly preferable in interactive graphics design. Applying transformation on each point of the curve is equivalent to applying transformation first to control points and redrawing the curve.



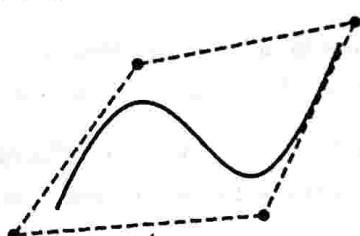
(a) Interpolation



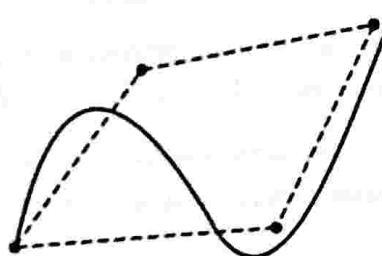
(b) Approximation

Fig. 8.1.1 : Interpolation v/s Approximation

- Many curves satisfy the convex hull property. The **convex hull** is the enclosed polygon which passes through the outermost control points. We say the curve is satisfying convex hull property if the entire curve lies within the convex hull.



(a) Curve satisfies convex hull property



(b) The curve does not satisfy convex hull property

Fig. 8.1.2 : Convex hull

- Curve having convex hull property indicates that the curve grows smoothly within the convex hull. Curves which do not satisfy the convex hull property may oscillate arbitrarily.
- Clipping the curve satisfying convex hull property is easy. Trivial acceptance and rejection conditions are easy to evaluate for them. If bounding box of the convex hull is inside clipping region, no need to clip the curve. And if a bounding box is completely outside clipping window, clip the entire curve.

8.1.3 Continuity

Q. What do you understand by Degree of Continuity w.r.t. Curve Generation?

MU - Dec. 19, 2 Marks

- When we estimate long curve by joining small curve segments, smoothness at joining point is expected. While traversing a connected curve, transition from one curve segment to other segment should not be noticed. At join point, motion should not be disturbed.

To achieve such smooth joining, we can impose certain continuity conditions at joining point. There are two types of continuity conditions under consideration.

8.1.3.1 Geometric Continuity

- For geometric continuity, derivative at the joining point of two curves should be proportional to each other. The i^{th} order geometric continuity is denoted by G^i . Let us say, $P(t)$ and $Q(t)$ are the two curve segments.
- **Zero order geometric continuity (G^0)** : If last control point of first segment and first control point of second segment is common, curve has zero order geometric continuity. Constraints for G^0 continuity are very relaxed. If two curves are connected, they are G^0 continuous. Condition for being curves zero order continuous : $P(1) = Q(0)$.
- **First order geometric continuity (G^1)** : If first order derivative of both curves are proportional at joining point, we say curves have first order geometric continuity. Condition for being curve first order continuous : $P'(1) = kQ'(0)$.
- **Second order geometric continuity (G^2)** : If first and second order derivative of both curves are proportional at joining point, we say curves have second order geometric continuity. Condition for being curve second order continuous : $P''(1) = kQ''(0)$

8.1.3.2 Parametric Continuity

- We can verify or derive parametric continuity by matching the direction and magnitude of derivative at the joining point of two curve segments. In parametric condition, derivative at the joining point of two curve segment should be same. The i^{th} order parametric continuity is denoted by C^i .
- **Zero order parametric continuity (C^0)** : Conditions for zero order continuity is identical for geometric and parametric continuity. Condition for being curves first order continuous : $P(1) = Q(0)$.
- **First order geometric continuity (C^1)** : If first order derivative of both curves are same at joining point, we say curves have first order parametric continuity. Condition for being curves second order continuous : $P'(1) = Q'(0)$.
- **Second order geometric continuity (C^2)** : If first and second order derivative of both curves are same at joining point, we say curve has second order parametric continuity. Condition for being curves second order continuous : $P''(1) = Q''(0)$.
- From above observations, geometry continuity is implicit in parametric continuity. Curve is also G^i continuous if it is C^i continuous. However converse is not true.



(a) Zero order continuity...

(b) First order continuity

(c) Second order continuity

Fig. 8.1.3 : Continuity of curve



- Parametric continuity provides more smoothness at join compare to geometric continuity. First order continuity is used in digitizing drawing. Second order derivative is used to specify smooth transitions like camera motion and animation path. Higher order derivatives might be used to exploit even higher smoothness like shading.

8.1.4 Bezier Curve

Q. Explain what is meant by the Bezier curve? State the various properties of Bezier curve.

MU - May 18, May 19, 10 Marks

Q. Explain the Bezier curve with its properties and construct.

MU - Dec. 18, Dec. 19, 10 Marks

Q. What do you understand by Control points, Local and Global control w.r.t. Curve Generation?

MU - Dec. 19, 3 Marks

- Bezier curve was first invented by Pierre Bezier. Due to its nice properties, Bezier curves are widely used in engineering designs and CAD packages.
- The curve is specified by a set of control points. Position of control points controls the shape of the curve. Fitting a single curve of the higher degree to all the control points is difficult. Normally, the big curve is approximated by joining piecewise polynomials of lower degree, which approximates few control points only.
- Local control : Repositioning one control point changes the entire curve.
- Global control : Repositioning one control point does changes the entire curve.
- With cubic polynomials, Bezier curve can approximate any number of control points. In the Bezier curve, the degree of polynomial depends on a number of control points used to generate curve segment.
- In the Bezier curve, the degree of the polynomial is always one less than a number of control points. Fig. 8.1.4 shows Bezier curves with three, four and five control points having a degree of polynomial two, three and four, respectively.

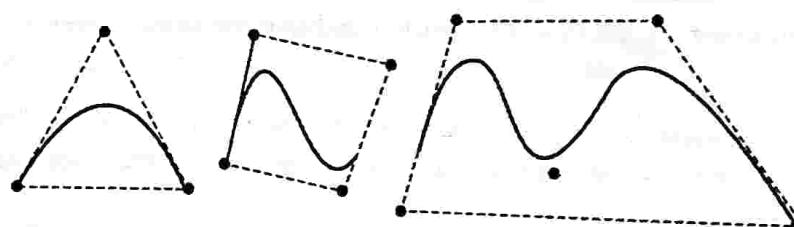


Fig. 8.1.4 : Bezier curves of degree 2, 3 and 4 with their convex hull

With $n + 1$ control points, parametric equation of Bezier curve approximating points p_0 to p_n ,

$$P(t) = \sum_{i=0}^n p_i \cdot BEZ_{i,n}(t), \quad 0 \leq t \leq 1$$

Blending functions of Bezier curves are directly derived from **Bernstein polynomials**.

$$BEZ_{i,n}(t) = C(n, i) t^i (1-t)^{n-i}$$

$C(n, i)$ is the binomial coefficient :

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

Bezier curve equation for each direction is written as,

$$x(t) = \sum_{i=0}^n x_i \cdot BEZ_{i,n}(t), 0 \leq t \leq 1$$

$$y(t) = \sum_{i=0}^n y_i \cdot BEZ_{i,n}(t), 0 \leq t \leq 1$$

$$z(t) = \sum_{i=0}^n z_i \cdot BEZ_{i,n}(t), 0 \leq t \leq 1$$

Properties of Bezier Curves

- Degree of the curve is one less than a number of control points
- Always interpolates first and last control points and approximates remaining two.
- The slope of the derivative at the beginning is along the line joining the first two points and slope of the derivative at the end is along the line joining last two points.
- Bezier curve always satisfies the convex hull property.
- At any parameter value t , sum all four Bezier blending function is always 1, i.e.,

$$\sum_{i=0}^n BEZ_{i,n}(t) = 1$$

- Polynomial smoothly follows the control points without much oscillation.
- Bezier curves do not have local control; repositioning one control point changes the entire curve.
- The curve is invariant under affine transformation.
- The basis functions are real.
- The curve exhibits variation diminishing property, i.e. any line intersects the Bezier curve at most as often as that line intersects the polygon interpolating control points.
- Bezier curve can fit any number of control points.
- Reversing the order of control points yield the same Bezier curve.

Applications of Bezier Curves

Bezier curve is used in many design applications. Bezier curves are used to model shapes and surfaces of automobiles, engineering, architectures, and aeronautics objects. Few of the applications are listed where the Bezier curve is being used to model the shapes.

- | | |
|------------------------------|------------------------------------|
| ○ Computer graphics | ○ Computer animation |
| ○ Engineering design | ○ CAD packages |
| ○ Modelling | ○ Graphics packages like photoshop |
| ○ Designing automobile parts | |



Cubic Bezier Curves

- Cubic Bezier curve is important to discuss. Cubic Bezier curves are the bridge between computation cost and smoothness.
- They are very much flexible and estimate shape nicely. We should specify four control points to generate a cubic Bezier curve.
- Blending functions for cubic Bezier curve is derived by putting $n = 3$ in the equation of Bezier curve,

$$BEZ_{0,3}(t) = (1-t)^3$$

$$BEZ_{1,3}(t) = 3t(1-t)^2$$

$$BEZ_{2,3}(t) = 3t^2(1-t)$$

$$BEZ_{3,3}(t) = t^3$$

- Fig. 8.1.5 shows the plot of blending functions over parameter range $0 \leq t \leq 1$. For $t = 0$, only first blending function is non zero and it has value 1. And at $t = 1$, only last blending function is non zero and its value is 1.

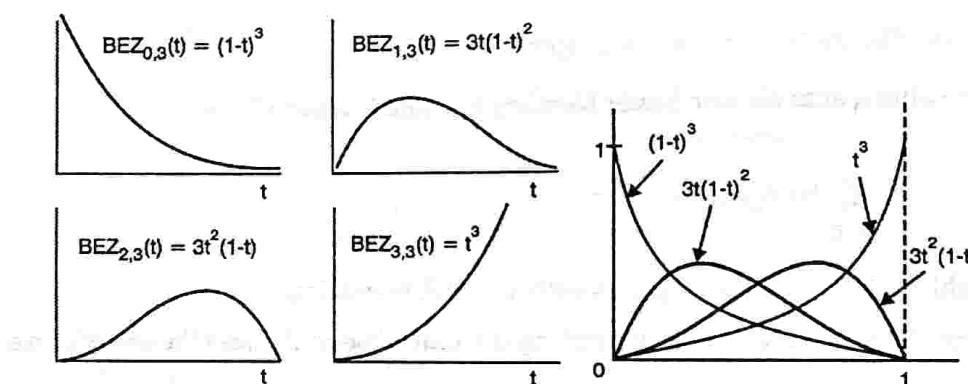


Fig. 8.1.5 : Bezier blending functions

- Thus Bezier curve always interpolates endpoints. For the rest of the values of t , all blending functions contribute to the shape of the curve.
- Blending functions $BEZ_{1,3}$ and $BEZ_{2,3}$ are maximum at $t = 1/3$ and $t = 2/3$ respectively. At any value of t , the summation of Bezier blending function is always 1.

Matrix Representation

We can derive a matrix representation of a cubic Bezier curve using blending function representation as below :

$$\begin{aligned} P(t) &= \sum_{i=0}^n p_i \cdot BEZ_{i,n}(t), 0 \leq t \leq 1 \\ &= p_0 \cdot BEZ_{0,3}(t) + p_1 \cdot BEZ_{1,3}(t) + p_2 \cdot BEZ_{2,3}(t) + p_3 \cdot BEZ_{3,3}(t) \\ &= p_0(1-t)^3 + p_1 3t(1-t)^2 + p_2 3t^2(1-t) + p_3 t^3 \\ &= (1 - 3t + 3t^2 - t^3)p_0 + (3t - 6t^2 + 3t^3)p_1 + (3t^2 - 3t^3)p_2 + t^3 p_3 \end{aligned}$$

$$P(t) = [t^3 \ t^2 \ t^1] \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$P(t) = T \cdot M_{\text{Bez}} \cdot G_{\text{Bez}}$$

Where, M_{Bez} and G_{Bez} represent Bezier basis matrix and Bezier geometric matrix respectively.

Limitations of the Bezier Curve

- Bezier curve approximates the control points so it cannot be used in the applications which require interpolating curve.
- Bezier curve does not have local control i.e. changing the position of one control point influences the entire curve.
- Degree of the curve depends on a number of the control point i.e. adding more control points increases the smoothness of the curve at the cost of processing higher degree polynomials.

Subdivision Method

- Another method to generate a Bezier curve is a subdivision method. This method generates a Bezier curve by midpoint calculations. Let us derive the Bezier curve defined by control points P_1, P_2, P_3 and P_4 .
- Join consecutive control points using straight lines. Suppose midpoint of lines joining points P_1P_2, P_2P_3 and P_3P_4 are L_2, H and R_3 respectively. Naming conventions are chosen to simplify the understanding of computation.

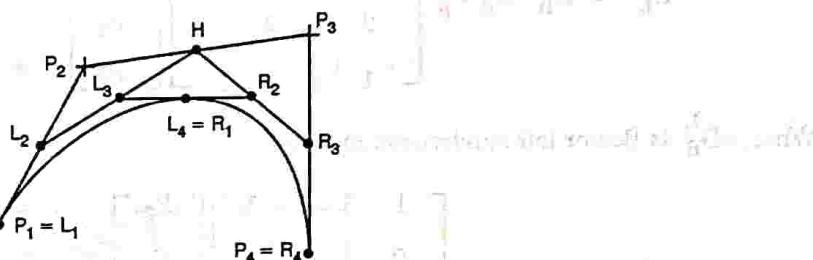


Fig. 8.1.6 : Bezier curve generation using subdivision method

- Join points L_2 and H and name its midpoint L_3 . Similarly, join points H and R_3 and name its midpoint R_2 . Join points L_3 and R_2 and assign label $L_4 = R_1$ to its midpoint ($L_4 = R_1$).
- Recursively draw two Bezier curve segments between control points (L_1, L_2, L_3, L_4) and (R_1, R_2, R_3, R_4) respectively. Both these curve segments are independent Bezier curves.
- Numbers of recursive calls decide the smoothness of the curve. We can derive the matrix representation for this subdivision as follows.

From Fig. 8.1.6,

$$L_1 = P_1 = \frac{8P_1}{8}$$



$$L_2 = \frac{P_1 + P_2}{2} = \frac{4P_1 + 4P_2}{8}$$

$$H = \frac{P_2 + P_3}{2}$$

$$L_3 = \frac{L_2 + H}{2} = \frac{\frac{P_1 + P_2}{2} + \frac{P_2 + P_3}{2}}{2}$$

$$L_3 = \frac{P_1 + 2P_2 + P_3}{4} = \frac{2P_1 + 4P_2 + 2P_3}{8}$$

$$R_4 = P_4 = \frac{8P_4}{8}$$

$$R_3 = \frac{P_3 + P_4}{2} = \frac{4P_3 + 4P_4}{8}$$

$$R_2 = \frac{H + R_3}{2} = \frac{\frac{P_2 + P_3}{2} + \frac{P_3 + P_4}{2}}{2} = \frac{P_2 + 2P_3 + P_4}{4} = \frac{2P_2 + 4P_3 + 2P_4}{8}$$

$$L_4 = R_1 = \frac{L_3 + R_2}{2} = \frac{\frac{P_1 + 2P_2 + P_3}{4} + \frac{P_2 + 2P_3 + P_4}{4}}{2} = \frac{P_1 + 3P_2 + 3P_3 + P_4}{8}$$

Bezier left subdivision matrix is given as,

$$G_B^L = D_B^L \cdot G_B = \frac{1}{8} \begin{bmatrix} 8 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 2 & 4 & 2 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Where, D_B^L is Bezier left subdivision matrix

$$G_B^R = D_B^R \cdot G_B = \begin{bmatrix} 1 & 3 & 3 & 1 \\ 0 & 2 & 4 & 2 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Where, D_B^R is Bezier right subdivision matrix.

Ex. 8.1.1 : Given $B_0 [1, 1]$, $B_1 [2, 3]$, $B_2 [4, 3]$ and $B_3 [3, 1]$ the vertices of a Bezier polygon determine the points on the Bezier curve.

Soln. :

The curve is defined by control points B_0, B_1, B_2, B_3 . The curve passes through the endpoints $(1, 1)$ and $(3, 1)$.

Matrix representation of the Bezier curve is,

$$Q(t) = T \cdot M_B \cdot G_B = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ 4 & 3 \\ 3 & 1 \end{bmatrix} = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -4 & 0 \\ 3 & -6 \\ 3 & 6 \\ 1 & 1 \end{bmatrix}$$

∴ Equation of Bezier curve would be,

$$Q(t) = [-4t^3 + 3t^2 + 3t + 1 \quad -6t^2 + 6t + 1]$$

Let us find few points on curve for given parameter value,

$$\begin{aligned} t = 0 \Rightarrow Q(0) &= [-4(0)^3 + 3(0)^2 + 3(0) + 1 \quad -6(0)^2 + 6(0) + 1] \\ &= [1 \quad 1] \end{aligned}$$

$$\begin{aligned} t = 0.2 \Rightarrow Q(0.2) &= [-4(0.2)^3 + 3(0.2)^2 + 3(0.2) + 1 \quad -6(0.2)^2 + 6(0.2) + 1] \\ &= [-0.032 + 0.12 + 0.6 + 1 \quad -0.24 + 1.2 + 1] \\ &= [1.688 \quad 1.96] \end{aligned}$$

$$\begin{aligned} t = 0.5 \Rightarrow Q(0.5) &= [-4(0.5)^3 + 3(0.5)^2 + 3(0.5) + 1 \quad -6(0.5)^2 + 6(0.5) + 1] \\ &= [-0.5 + 0.75 + 1.5 + 1 \quad -1.5 + 3 + 1] \\ &= [2.75 \quad 2.5] \end{aligned}$$

$$\begin{aligned} t = 0.8 \Rightarrow Q(0.8) &= [-4(0.8)^3 + 3(0.8)^2 + 3(0.8) + 1 \quad -6(0.8)^2 + 6(0.8) + 1] \\ &= [-2.048 + 1.92 + 2.4 + 1 \quad -3.84 + 4.8 + 1] \\ &= [3.272 \quad 1.96] \end{aligned}$$

$$\begin{aligned} t = 1.0 \Rightarrow Q(1.0) &= [-4(1.0)^3 + 3(1.0)^2 + 3(1.0) + 1 \quad -6(1.0)^2 + 6(1.0) + 1] \\ &= [-4 + 3 + 3 + 1 \quad -6 + 6 + 1] \\ &= [3 \quad 1] \end{aligned}$$

Value of t	Point on curve
t = 0.0	[1 1]
t = 0.2	[1.688 1.96]
t = 0.5	[2.75 2.5]
t = 0.8	[3.272 1.96]
t = 1.0	[3 1]

Ex. 8.1.2 : Find the equation for the Bezier curve, which passes through control points (0, 0) and (-3, 2) and controlled by (6, 4) and (3, 1). Also find points on curve for t = 0, 0.4, 0.8, 1.

Soln. : Assume that curve is defined by control points p_0, p_1, p_2 and p_3 . The curve passes through the endpoints (0, 0) and (-3, 2).



So, $p_0 = (0, 0)$ and $p_3 = (-3, 2)$. Let us consider $p_1 = (6, 4)$ and $p_2 = (3, 1)$

Matrix representation of Bezier curve is,

$$Q(t) = T \cdot M_B \cdot G_B = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 6 & 4 \\ 3 & 1 \\ -3 & 2 \end{bmatrix} = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} 6 & 11 \\ -27 & -21 \\ 18 & 12 \\ 0 & 0 \end{bmatrix}$$

\therefore Equation of Bezier curve would be,

$$Q(t) = [6t^3 - 27t^2 + 18t \quad 11t^3 - 21t^2 + 12t]$$

Let us find point on curve for given parameter value,

$$\begin{aligned} t = 0 \Rightarrow Q(0) &= [6(0) - 27(0) + 18(0) \quad 11(0) - 21(0) + 12(0)] \\ &= [0 \quad 0] \end{aligned}$$

$$\begin{aligned} t = 0.4 \Rightarrow Q(0.4) &= [6(0.4)^3 - 27(0.4)^2 + 18(0.4) \quad 11(0.4)^3 - 21(0.4)^2 + 12(0.4)] \\ &= [0.384 - 4.32 + 7.2 \quad 0.704 - 3.36 + 4.8] \\ &= [3.264 \quad 2.144] \end{aligned}$$

$$\begin{aligned} t = 0.8 \Rightarrow Q(0.8) &= [6(0.8)^3 - 27(0.8)^2 + 18(0.8) \quad 11(0.8)^3 - 21(0.8)^2 + 12(0.8)] \\ &= [5.632 - 17.28 + 14.4 \quad 5.632 - 13.44 + 9.6] \\ &= [2.752 \quad 1.792] \end{aligned}$$

$$\begin{aligned} t = 1 \Rightarrow Q(1) &= [6(1)^3 - 27(1)^2 + 18(1) \quad 11(1)^3 - 21(1)^2 + 12(1)] \\ &= [6 - 27 + 18 \quad 11 - 21 + 12] = [-3 \quad 2] \end{aligned}$$

Value of t	Point on curve
t = 0.0	[0 0]
t = 0.4	[3.264 2.144]
t = 0.8	[2.752 1.792]
t = 1.0	[-3 2]

Ex. 8.1.3 : Construct the Bezier curve of order three with control points $P_1(0, 0)$, $P_2(1, 3)$, $P_3(4, 2)$ and $P_4(2, 1)$. Generate at least five points on the curve.

Soln. :

The curve passes through the endpoints $(0, 0)$ and $(2, 1)$.

Matrix representation of the Bezier curve is,

$$Q(t) = T \cdot M_B \cdot G_B = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 3 \\ 4 & 2 \\ 2 & 1 \end{bmatrix}$$

$$= [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -7 & 4 \\ 6 & -12 \\ 3 & 9 \\ 0 & 0 \end{bmatrix}$$

∴ Equation of Bezier curve would be,

$$Q(t) = [-7t^3 + 6t^2 + 3t \quad 4t^3 - 12t^2 + 9t]$$

We have to find at least five points on the curve. Let us find point on curve for parameter step value 0.2.

$$t = 0.0 \Rightarrow Q(0.0)$$

$$= [-7(0.0)^3 + 6(0.0)^2 + 3(0.0) \quad 4(0.0)^3 - 12(0.0)^2 + 9(0.0)]$$

$$= [0 + 0 + 0 \quad 0 - 0 + 0]$$

$$= [0 \quad 0]$$

$$t = 0.2 \Rightarrow Q(0.2)$$

$$= [-7(0.2)^3 + 6(0.2)^2 + 3(0.2) \quad 4(0.2)^3 - 12(0.2)^2 + 9(0.2)]$$

$$= [-0.056 + 0.24 + 0.6 \quad 0.032 - 0.48 + 1.8]$$

$$= [0.784 \quad 1.352]$$

$$t = 0.4 \Rightarrow Q(0.4)$$

$$= [-7(0.4)^3 + 6(0.4)^2 + 3(0.4) \quad 4(0.4)^3 - 12(0.4)^2 + 9(0.4)]$$

$$= [-0.448 + 0.96 + 1.2 \quad 0.256 - 1.92 + 3.6]$$

$$= [1.712 \quad 1.424]$$

$$t = 0.6 \Rightarrow Q(0.6)$$

$$= [-7(0.6)^3 + 6(0.6)^2 + 3(0.6) \quad 4(0.6)^3 - 12(0.6)^2 + 9(0.6)]$$

$$= [1.512 + 2.16 + 1.8 \quad 0.864 - 4.32 + 5.4]$$

$$= [5.472 \quad 1.944]$$

$$t = 0.8 \Rightarrow Q(0.8)$$

$$= [-7(0.8)^3 + 6(0.8)^2 + 3(0.8) \quad 4(0.8)^3 - 12(0.8)^2 + 9(0.8)]$$



$$= [-3.584 + 3.84 + 2.4 \quad 2.048 - 7.68 + 7.2]$$

$$= [2.656 \quad 1.568]$$

$$t = 1.0 \Rightarrow Q(1.0)$$

$$= [-7(1.0)^3 + 6(1.0)^2 + 3(1.0) \quad 4(1.0)^3 - 12(1.0)^2 + 9(1.0)]$$

$$= [-7 + 6 + 3 \quad 4 - 12 + 9]$$

$$= [2 \quad 1]$$

Value of t	Point on curve
t = 0.0	[0 0]
t = 0.2	[0.784 1.352]
t = 0.4	[1.712 1.424]
t = 0.6	[5.472 1.944]
t = 0.8	[2.656 1.568]
t = 1.0	[2 1]

Ex. 8.1.4 : Given four control points (10, 10), (15, 15), (20, 15) and (30, 10), find the points to plot Bezier curve by using step size as 0.2.

Soln. :

Assume that curve is defined by control points p_0, p_1, p_2 and p_3 . The curve passes through the endpoints (10, 10) and (30, 10),

So, $p_0 = (10, 10)$ and $p_3 = (30, 10)$. Let us consider $p_1 = (15, 15)$ and $p_2 = (20, 15)$

Matrix representation of the Bezier curve is,

$$Q(t) = T \cdot M_B \cdot G_B$$

$$= [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 10 & 10 \\ 15 & 15 \\ 20 & 15 \\ 30 & 10 \end{bmatrix} = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} 5 & 0 \\ 0 & -15 \\ 15 & 15 \\ 10 & 10 \end{bmatrix}$$

\therefore Equation of Bezier curve would be,

$$Q(t) = [5t^3 + 15t^2 + 10 \quad -15t^2 + 15t + 10]$$

Let us find point on curve for parameter step value 0.2.

$$t = 0.0 \Rightarrow Q(0.0)$$

$$= [5(0.0)^3 + 15(0.0)^2 + 10 \quad -15(0.0)^2 + 15(0.0) + 10]$$

$$= [0 + 0 + 10 \quad -0 + 0 + 10]$$

$$= [10 \quad 10]$$

$$t = 0.2 \Rightarrow Q(0.2)$$

$$= [5(0.2)^3 + 15(0.2) + 10 \quad - 15(0.2)^2 + 15(0.2) + 10]$$

$$= [0.04 + 3 + 10 \quad - 0.6 + 3 + 10]$$

$$= [13.04 \quad 12.4]$$

$$t = 0.4 \Rightarrow Q(0.4)$$

$$= [5(0.4)^3 + 15(0.4) + 10 \quad - 15(0.4)^2 + 15(0.4) + 10]$$

$$= [0.32 + 6 + 10 \quad - 2.4 + 6 + 10]$$

$$= [16.32 \quad 13.6]$$

$$t = 0.6 \Rightarrow Q(0.6)$$

$$= [5(0.6)^3 + 15(0.6) + 10 \quad - 15(0.6)^2 + 15(0.6) + 10]$$

$$= [1.08 + 9 + 10 \quad - 5.4 + 9 + 10]$$

$$= [20.08 \quad 13.6]$$

$$t = 0.8 \Rightarrow Q(0.8)$$

$$= [5(0.8)^3 + 15(0.8) + 10 \quad - 15(0.8)^2 + 15(0.8) + 10]$$

$$= [2.56 + 12 + 10 \quad - 9.6 + 12 + 10]$$

$$= [24.56 \quad 9.4]$$

$$t = 1.0 \Rightarrow Q(1.0)$$

$$= [5(1)^3 + 15(1) + 10 \quad - 15(1)^2 + 15(1) + 10]$$

$$= [5 + 15 + 10 \quad - 15 + 15 + 10]$$

$$= [30 \quad 10]$$

Value of t	Point on curve
$t = 0.0$	[10 10]
$t = 0.2$	[13.04 12.40]
$t = 0.4$	[16.32 13.60]
$t = 0.6$	[20.08 13.60]
$t = 0.8$	[24.56 9.40]
$t = 1.0$	[30 10]



8.1.5 B-Spline Curve

Q. Explain what is meant by B-Spline curve? State the various properties of B-Spline curve. **MU - Dec. 19, 10 Marks**

- In the Bezier curve, the degree of the polynomial is always one less than the number of control points. So we cannot have more than four points for cubic Bezier curves.
- Furthermore, change in the single control point has a global effect on the Bezier curve. B-Spline curves are the most widely used class of curves for approximating the shape due to its following properties :
 - o Degree of a polynomial is independent of a number of control points (which is not true in case of Bezier curve).
 - o They have local control over the curve and surfaces.
 - o However, derivation and generation of B-spline are more complex than Bezier curves.
 - o The B-spline curve with $(n + 1)$ control points is expressed in the form of blending function as,

$$P(t) = \sum_{i=0}^n p_k \cdot B_{i,d}(t), t_{\min} \leq t \leq t_{\max} \text{ and } 2 \leq d \leq n + 1$$

- For Bezier curve, the range of t was always between 0 and 1. In the B-spline curve, t depends on types of B-spline.
- B-Spline blending functions $B_{i,d}$ are polynomials with degree $d - 1$. Degree d can take any integer value between 2 and the number of control points.
- We can achieve local control by defining the blending function on subinterval of the total range of parameter t .
- Blending functions of Bezier curve are described by Bernstein polynomials, whereas blending functions for B-spline curve are defined using Cox-deBoor recursive formula as shown below :

$$B_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,d}(t) = \frac{t - t_i}{t_{i+d-1} - t_i} B_{i,d-1}(t) + \frac{t_{i+d} - t}{t_{i+d} - t_{i+1}} B_{i+1,d-1}(t)$$

- Blending functions are defined over d sub-intervals of the total range of t . So change in one control point can affect maximum d piecewise small curves, it does not have a global effect on the curve.
- The selected set of subinterval t_i is called **knot vector**. We can choose any value for t_i in monotonically increasing order, i.e. $t_i \leq t_{i+1}$. However, the value of t_{\min} and t_{\max} is controlled by a number of control points, parameter d and how we set up a knot vector.

Properties of B-Spline Curve

- Degree of polynomial does not depend on the number of control points.
- B-Spline curves do not interpolate any of the control points.
- The generated polynomial has degree $d - 1$ with inherent C^{d-2} parametric continuity.
- The curve has $n + 1$ blending function if it is described with $n + 1$ control points

- Size of knot vector is $n + d + 1$.
- The entire range of parameter t is divided into $(n + d)$ subintervals.
- If the knot vector is defined as $\{t_0, t_1, t_2, \dots, t_{n+d}\}$, B-spline curve spans only in the interval t_{d-1} to t_{n-1} .
- Each blending function $B_{i,d}$ spans over d subinterval of the entire range of t , starting from knot value t_i .
- B-Spline allows local control.
- C^2 continuity is inherent in B-Spline.
- Each B-spline curve segment is affected by d control points.
- Change in one control point influence shape of at most d curve sections.
- B-Spline curve satisfies the property of convex hull, so they are tightly bound within the control point convex hull boundary.
- For any value of t in the range t_{d-1} to t_{n+1} , the sum of all blending function is 1, i.e.,

$$\sum_{i=0}^n B_{i,d}(t) = 1$$

According to the specification of knot vector, B-spline curves are classified into three categories :

1. Uniform B-Splines
2. Open uniform B-splines
3. Non-uniform B-Splines

1. Uniform Periodic B-Spline

- In uniform B-Splines, knot values in knot vector are equispaced, i.e. the difference between two consecutive knot values is constant. For example $\{-3, -2, -1, 0, 1, 2, 3, 4\}$ is uniform knot vector. Knot values are often normalized such that they fall in range 0 to 1, for example $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$.
- A more convenient way is to represent the knot vector with integer knot values, starting from zero, i.e. $\{0, 1, 2, 3, 4, 5, 6, 7\}$.
- Blending functions of uniform B-spline are periodic and successive blending function is translated/shifted version of the previous one.

$$B_{i,d}(t) = B_{i+1,d}(t + \Delta t) = B_{i+2,d}(t + 2\Delta t) = \dots$$

- Where Δt is the difference between two successive knot values. Blending functions for uniform B-spline are shown in Fig. 8.1.7.

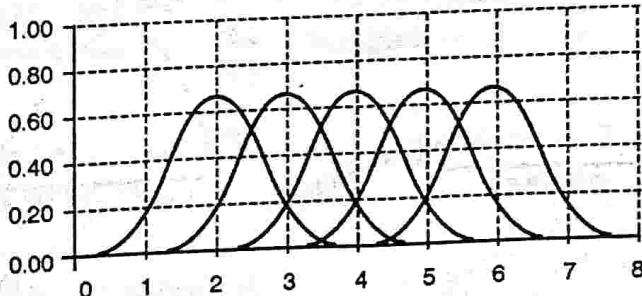


Fig. 8.1.7 : Blending functions for uniform B-Spline

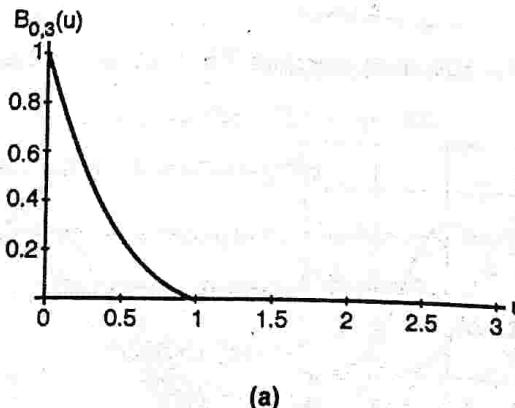


2. Open Uniform B-Spline

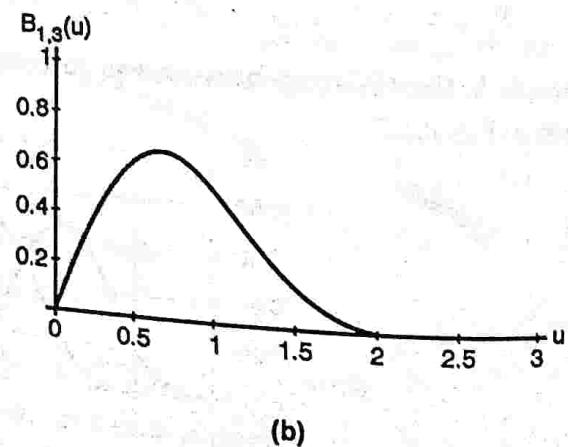
- Open uniform B-spline is the bridge between uniform and non-uniform splines. Sometimes it is considered as a special case of uniform B-spline; sometimes it is considered as a special case of non-uniform B-Spline.
- It differs from uniform spline in the specification of a knot vector. Like uniform splines, knot vector in open uniform also has uniform spacing between successive knot values except first and last. End knot values are repeated d times.
 - o For $d = 2$ and $n = 4$, size of knot vector is $n + 1 + d = 7$. First and last knot value is repeated $d = 2$ times.
So, starting with 0, knot vector is specified as, {0, 0, 1, 2, 3, 4, 4}
 - o For $d = 4$ and $n = 5$, size of knot vector is $n + 1 + d = 10$. First and last knot value is repeated $d = 4$ times.
So, starting with 0, knot vector is specified as, {0, 0, 0, 0, 1, 2, 3, 3, 3, 3}
- We can normalize the range of knot value between 0 and 1.
 - o For $d = 2$ and $n = 4$, {0, 0, 0.25, 0.5, 0.75, 1, 1}
 - o For $d = 4$ and $n = 5$, {0, 0, 0, 0, 0.33, 0.66, 1, 1, 1, 1}
- We can generate integer valued knot vector for given n and d using following formula :

$$t_i = \begin{cases} 0 & \text{for } 0 \leq i < d \\ i - d + 2 & \text{for } d \leq i \leq n \\ n - d + 2 & \text{for } i > n \end{cases}$$

- The first knot has value 0 and last has value $n - d + 2$.
- Characteristics of the open B-spline curve are very much similar to the Bezier curve. Open B-spline reduces to Bezier curve when the degree of the polynomial is one less than the control point, i.e. $d = n + 1$. For such cubic B-spline, $d = 4$, $n + d = 4$, $n + 1 = 4$, so size of knot vector is 8. In this case, knot vector would be {0, 0, 0, 0, 1, 1, 1, 1}.
- Blending functions for open uniform B-spline with $n = 4$ and $d = 3$ are shown in Fig. 8.1.8.



(a)



(b)

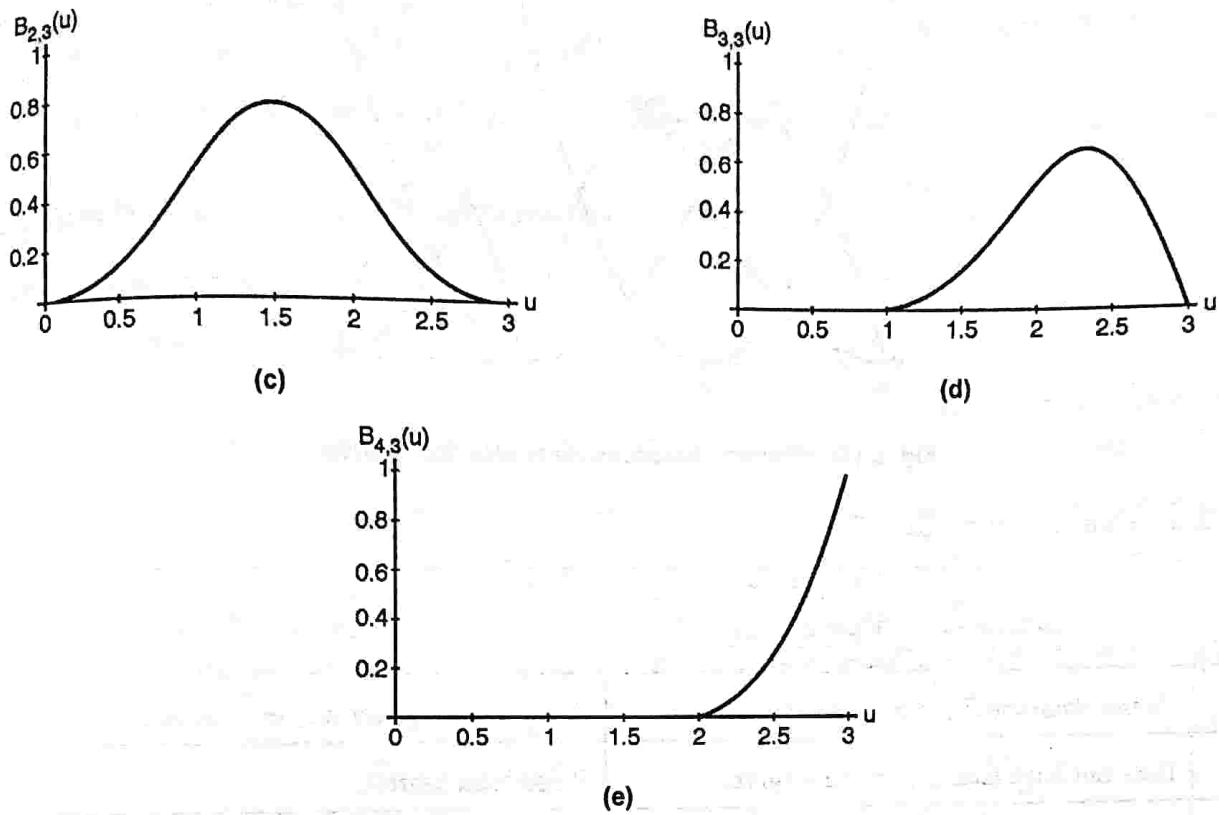


Fig. 8.1.8 : Blending functions for open uniform curve

3. Non-uniform B-spline

- Knot vector for this class of polynomial is least restrictive. Knot vector can repeat the knot values; spacing between successive knot values can be different. The only constraint is that knot value must be in monotonically increasing order.
- Examples are,
 - {0, 1, 2, 3, 4, 5}
 - {0, 1, 1, 2, 3, 4, 4, 5}
 - {0, 0.1, 0.4, 0.4, 0.7, 0.9, 1, 1}
- Such curves are most flexible in controlling the shape. Due to the unequal spacing of knot value, we get blending functions with the different shape at different intervals. By repeating the knot values, we can disturb the shape of the curve and even we can introduce a discontinuity in the curve. For each repeat, discontinuity of curve reduces by one. Derivation of blending functions of non-uniform B-spline is the same as uniform and open uniform curves.
- With the help of $n + 1$ control points, we can use Cox-deBoor recursive formula to find blending function with specified degree and knot vector. Matrix representation can be used to avoid recursive computation of Cox-deBoor equation.
- Blending function for non-uniform B-Spline is shown in Fig. 8.1.9.

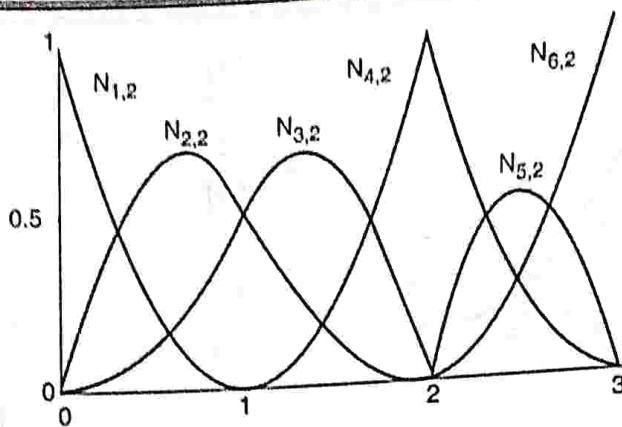


Fig. 8.1.9 : Blending functions for non-uniform curve

8.1.6 Bezier Vs. B-Spline Curve

Sr. No.	Bezier Curve Generation	B-Spline Curve Generation
1.	Passes from the first and last control point.	Does not interpolate any control point.
2.	Does not have local control on a curve.	It has local control.
3.	Changing one control point affects the position of all points on the curve.	Changing one control point affects only a few points on the curve.
4.	Degree of Bezier curve is determined by the number of control points.	Degree of the B-Spline curve is independent of the number of control points.
5.	Need special treatment to join two Bezier curves with c_2 continuity.	Continuity is inherent in the B-Spline curve.
6.	Less flexible.	More flexible.
7.	During piecewise approximation, adds 3 control point to generate a new curve segment.	During piecewise approximation, adds only 1 control point to generate a new curve segment.

8.2 Fractals

8.2.1 Fractal-Geometry

Q. Write a short note on Fractals.

MU - Dec. 18, 5 Marks

Q. Write a short note on Fractals.

MU - Dec. 19, 10 Marks

- Shapes like line, circle, ellipse, curves etc. are defined using Euclidean geometry or mathematical equation.
- Natural objects like plants, tree, mountain, waves etc. cannot be described by such mathematical notion.
- Natural shapes have many irregularities and self-similarities.

- Shapes generated using Euclidean geometry have a smooth surface. Euclidean geometry methods are not suitable for generating a realistic display of the natural scene.
- Natural objects are well described using fractal geometry methods - which are actually a procedure rather than an equation.

Fractal Objects Possess the following Two Properties :

- Infinite detail at every point.
- Self-similarity between object parts.
- Natural objects have infinite details; however, we should design a process which produces finite detail because on the computer we cannot display infinite detail.
- After certain levels of zoom in, Euclidean shapes lead to smooth drawing. While in the fractal object, if we keep zoom in, we continue to see the same detail as it appears in the original object. Fig. 8.2.1 describes the discussed concept.

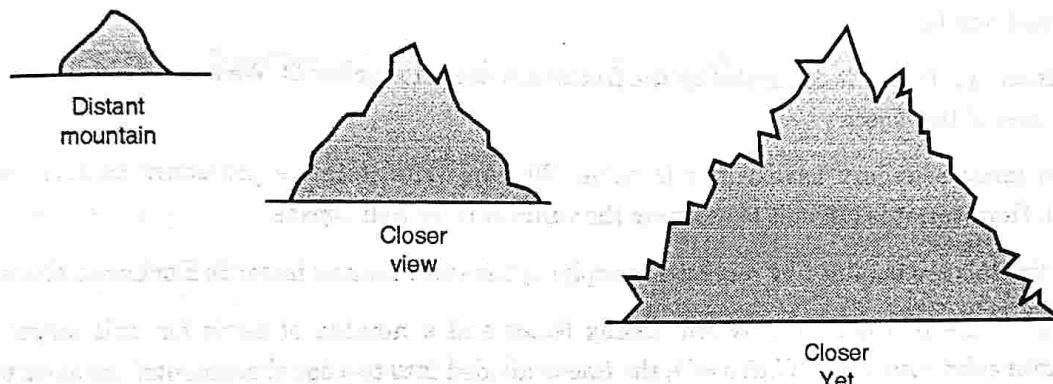


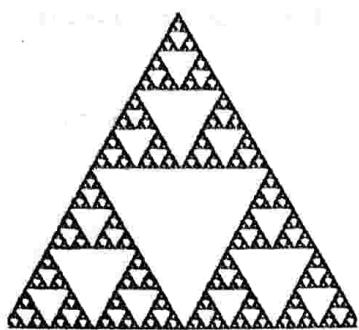
Fig. 8.2.1 : View of the mountain at different zoom level

8.2.2 Fractal Classification

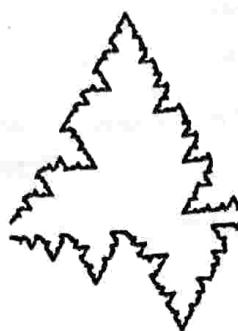
Fractals can be classified as follow :

1. **Self-similar fractals** : Such fractal exhibits the self-similarity in shape. They have components which are a scaled-down version of the object itself. We can construct the object by applying different scaling factors to all parts or by using the same scaling factor for all. If we use a random scaling factor to generate the subparts, a fractal is called **statistically self-similar**. Statistically self-similar fractals are used to construct shapes like tree, shrubs, plants etc.
2. **Self-affine fractals** : Such fractals use different scaling factors for all three directions. We can add random variation to get statistically self-affine fractals. This class of fractals is best suited to model terrain, water, clouds etc.
3. **Invariant fractals** : Nonlinear transformation is applied to construct such fractals. A self-squaring function in complex space produces self-squaring fractals like Mandelbrot. Self-inverse fractals are generated using the inversion procedure.

Example of each class is depicted in following Fig. 8.2.2 :



(a) Self-similar fractals



(b) Self-affine fractals

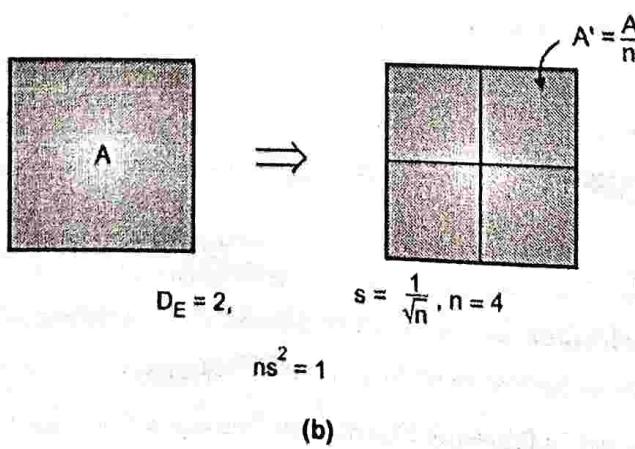
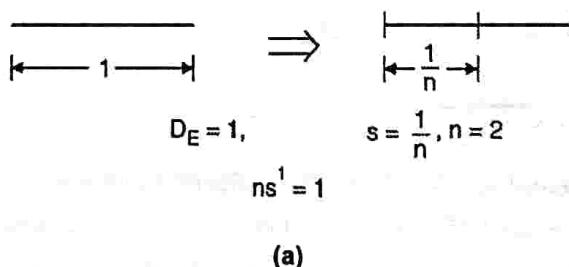


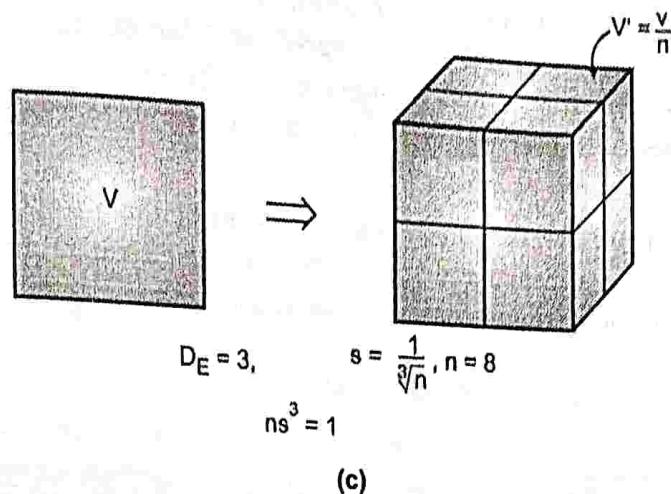
(c) Invariant fractal

Fig. 8.2.2 : Types of fractals

8.2.3 Fractal Dimension

- Amount of variation in object detail is described by a number called **fractal dimension**. Fractal dimension can be any real number.
- Detail variation in a fractal is controlled by the fractal dimension number D. We can consider D as a measure of the roughness of the object.
- Objects with jaggy boundary have larger D value. We can write iterative procedure controlling detail by specifying D. However, it is difficult to estimate the value of D for real objects.
- Self-similar fractals are obtained by recursively applying the same scaling factor to Euclidean shapes.
- Fig. 8.2.3 shows the relationship between scaling factor and a number of parts for unit length line, unit square and unit cube with $s = \frac{1}{2}$. With $s = \frac{1}{2}$, the line is divided into two equal segments, the square is divided into four equal parts of equal area and cube is divided into eight subcubes of equal volume.





**Fig. 8.2.3 : Subdivision of object with scaling factor $s = 1/2$ and Euclidean dimensions
((a) $D_E = 1$, (b) $D_E = 2$ and (c) $D_E = 3$)**

For these objects, the relationship between a number of subparts and the scaling factor is $n.s^{DE} = 1$. For self-similar objects,

$$n.s^D = 1$$

Solving the equation,

$$D = \frac{\ln n}{\ln (1/s)}$$

- D is called the fractal similarity dimension. For a self-similar object with different scaling factor for a different part, self-similarity dimension is obtained as,

$$\sum_{i=1}^n S_i^D = 1$$

Where S_i is the scaling factor for i^{th} subpart of the object.

- For complex shapes like curves and surfaces, it is difficult to estimate the characteristics of subparts of the shape. The large shape is approximated by subdivides the shape into smaller simpler shapes, that is called **topological covering**. For example, smooth curves can be approximated using a very small line segment. The surface can be approximated using small polygons like triangles.
- Covering methods are used to determine object properties like length, area, the volume of an object by summing these properties smaller objects - forming the parent object. Covering methods also help in determining fractal dimensions (D).
- Fractals are generated by applying transformation function repeatedly to the points in the region of interest. If $P(x, y, z)$ is the initial point, each iteration of transformation function generates detail as,

$$P_1 = F(P), P_2 = F(P_1), P_3 = F(P_2), \dots$$



- Transformation function can be applied to a set of points of selected base primitive. This primitive can be anything like line, arc, curve, surface etc. We can specify a deterministic or non-deterministic (random) process in each iteration. The transformation function can be either simple geometry transformation operations like translation, rotation, scaling, reflection etc, or it can be specified using non-linear transformation functions.
- The objects we create and display on the screen is generated using finite transformation operation. So they have finite dimensions. We cannot display the details which are smaller than the pixel. Details appearing on view depend on a number of transformations we applied. If we increase the number of transformation, more and more subtle details are added to the scene.

8.2.4 Koch Curve

Q. Write a short note on the Koch curve.

MU - May 18, Dec. 18, May 19, 10 Marks

- Koch curve begins with the straight line. It divides the line into three segments. The middle segment is replaced by two sides of an equilateral triangle with the apex pointing outside. Now we have four segments. This same process is repeated on these four segments. General procedure for generating the Koch curve can be stated as :
 1. Divide a straight line into three equal parts.
 2. Draw an equilateral triangle on the middle segment, pointing in an outward direction with a base of the middle segment.
 3. Remove the line middle line segment.



Fig. 8.2.4 : Initiator and generator for Koch curve

- This procedure is repeated again and again to achieve more and more smoothness on the curve.
- To construct, deterministic, self-similar fractals, we start with geometric shapes called the initiator. A subpart of the initiator is then repeatedly replaced by generator pattern.
- Fig. 8.2.4 shows the initiator and generator pattern for the Koch curve.
- Each line segment of initiator is replaced by four segments of the generator. Gradually, the length of initiator tends to infinity, which more and more details are added to curve.
- After each iteration, the curve length is increased by one third. The number of line segments increases by a factor of four after each iteration. Length of each newly created line segment is $1/3^{\text{rd}}$ of the original line segment. The scaling factor is $1/3$, so the fractal dimension is,

$$D = \frac{\ln(n)}{\ln(1/s)} = \frac{\ln(4)}{\ln(1/(1/3))} = \frac{\ln(4)}{\ln(3)} = 1.2619$$

- The total length of initiator increases by factor $4/3$ in each iteration. Thus, the length of the curve after n iteration would be $(4/3)^n$. Fig. 8.2.5 illustrate the length calculation of the Koch curve.

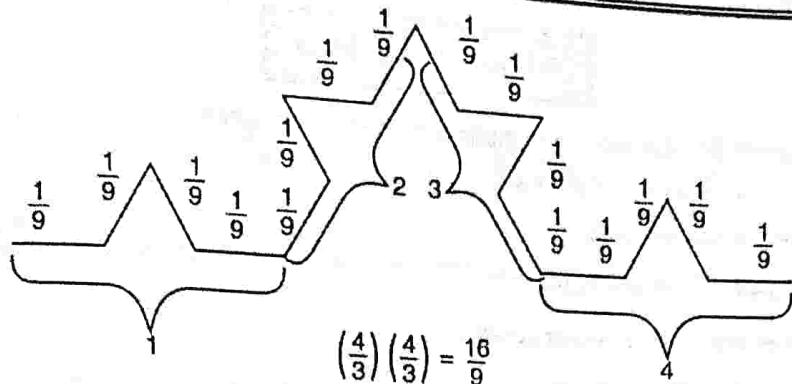


Fig. 8.2.5 : Length calculation after two iterations

- Fractals are non-ending patterns. Each iteration adds finer detail. Fig. 8.2.6 shows the procedure of the Koch curve generation.

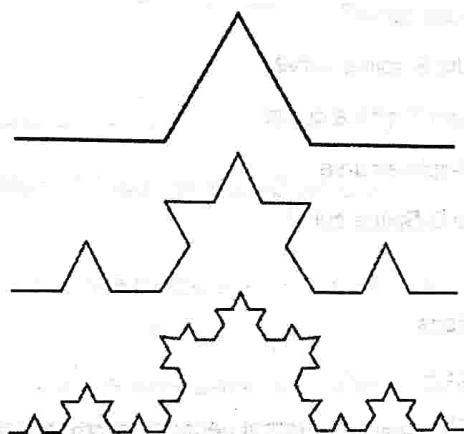


Fig. 8.2.6 : Generation of the Koch curve

8.2.5 Applications

Fractals are used to model the natural shapes. Some of the prominent application areas of fractals are listed here :

- Modelling natural structures like Geographic terrain, mountain, plant structure, clouds, vegetables etc.
- Study of the chaotic phenomenon
- Fractal art
- Space research
- Study of convergence of iterative processes
- Engineering and architecture
- Medical science
- Chemical processes
- Medical diagnostic images
- Fluid mechanics
- Image compression
- Telecommunication

**Review Questions**

- Q. 1** What is the Bezier curve? Give the blending functions for the Bezier curve.
- Q. 2** Enlist the properties of Bezier curve.
- Q. 3** State a few applications of Bezier curve.
- Q. 4** Write a short note on the Cubic Bezier curve.
- Q. 5** Give the matrix representation of the Bezier curve.
- Q. 6** What are the limitations of the Bezier curve?
- Q. 7** How to generate a Bezier curve using the subdivision method.
- Q. 8** Write a note on the B-spline curve.
- Q. 9** What are the properties of the B-Spline curve?
- Q. 10** Write a short note on uniform periodic B-spline curve.
- Q. 11** Write a short note on the open uniform B-spline curve.
- Q. 12** Write a short note on non-uniform B-spline curve.
- Q. 13** Differentiate the Bezier curve vs. the B-Spline curve.
- Q. 14** What do you mean by fractals?
- Q. 15** Write a short note on fractal dimensions.
- Q. 16** Write a short note on Hilbert's curve.
- Q. 17** What are the applications of fractals? Explain the fractal generation procedure.
- Q. 18** Explain the Koch curve with diagram.

CHAPTER 9

Module 6

Syllabus

Visible Surface Detection : Classification of Visible Surface Detection algorithm, Back Surface detection method, Depth Buffer method, Area Subdivision method

9.1 Introduction

- In the real scene, there may be many 3D objects placed randomly, completely or partially hiding some other objects.
- At any given viewing direction, some of the surfaces may not be visible. Identifying only visible surfaces from the scene is a major concern of computer graphics.
- For displaying a realistic scene, it is often necessary to render visible surfaces only. Many algorithms have been devised to find visible surfaces efficiently.
- There is a time-space trade-off in different algorithms. Some methods are quick with more memory requirement. Some are space-efficient with higher running time.
- Visible surface detection algorithms are also known as *hidden line* or *hidden surface removal* algorithms. Aim of such algorithms is to identify the visible lines or surfaces of a given set of object from a particular viewing direction.

9.2 Classification of Visible Surface Detection Algorithm

- Visible surface detection algorithms may be classified either as object space algorithms or Image-space algorithms.

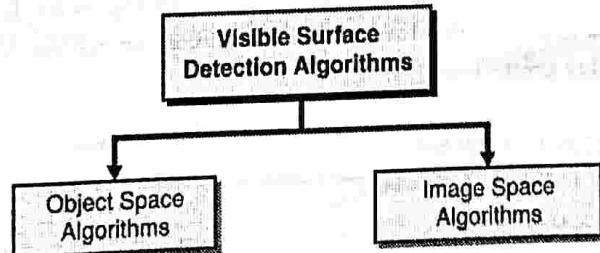


Fig. 9.2.1 : Classification of surface detection algorithms

1. Object Space Method

- Object space methods determine the visible surface of an object by comparing the entire object or part of an object with other objects within the scene. These algorithms operate in physical coordinates system.
- This class of algorithms is faster and performs less computation. Following methods belong to this category :
 - o Backface detection



- o Painter's algorithm
- o Robert's algorithm

2. Image Space Methods

- In the image space method, the visibility of the object is decided by pixel by pixel comparison of overlapping objects. The pixel of the nearest object to the viewer is selected for display. These algorithms operate in a screen coordinate system. Such algorithms are more precise but computationally intense. Following methods belong to this category :

- o Depth buffer method
- o Area subdivision method
- o Octree method
- o Scan line method
- o Ray tracing algorithm

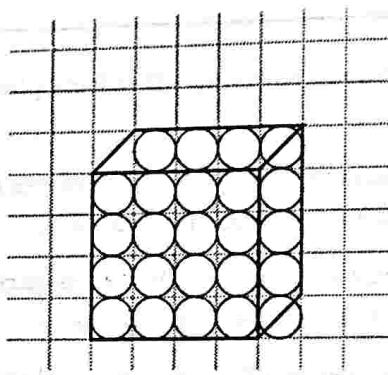
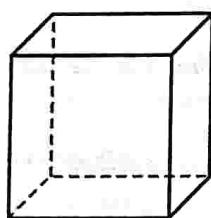


Fig. 9.2.2 : Object space v/s Image-space Approach

- Most of the visible surface detection algorithms are image space algorithms. However, the choice of algorithm depends on the application. Generally, object space algorithms are preferable.
- Performance of the algorithm can be improved with sorting and coherence property. All the objects in the scene are arranged from front to back in increasing order of their depth from the viewer.
- The object nearest to the viewer may hide the part of the scene, so we do not have to render a portion of hidden objects behind the visible front object.
- Thus, coherence property is used to take advantage of regularity in the scene. In the same way, the individual scan line can also be assumed to have some interval, either hidden or visible. Moreover, two consecutive scan lines also exhibit almost similar properties.

9.3 Back Surface Detection Method

- Sometimes visible surface detection is also referred to as hidden surface removal or back face detection. However, there is a subtle difference in these methods. It is the object space method.
- A visible surface detection algorithm decides which part of the scene is visible and projects it on screen.
- Whereas, back-face detection algorithms consider the entire object and keep removing hidden surfaces from the scene.

Former algorithms select visible surfaces to display, later category of algorithms removes hidden surfaces from the scene.

Hidden surfaces can be easily detected from the relationship between viewing direction and surface.

As shown in Fig. 9.3.1, if outward normal of surface and line of sight are same in this case surface would not be visible.

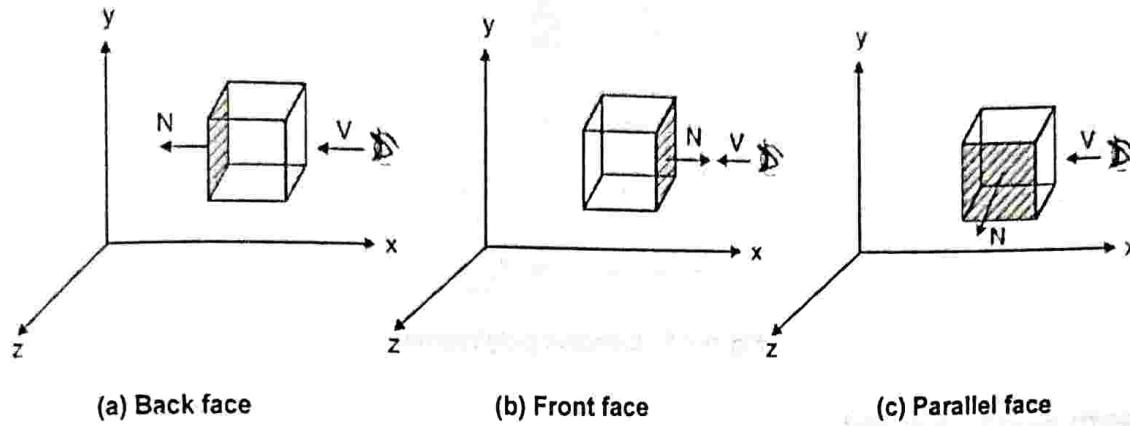


Fig. 9.3.1 : Cube surfaces and their normal

- As shown in Fig. 9.3.1(a), the shaded surface is parallel to YZ plane and it's outward normal (N) points in negative X-direction which is the same as the viewing direction (V).

So we consider,

$$N = (-1, 0, 0) \text{ and } V = (-1, 0, 0).$$

$$N \cdot V = (-1, 0, 0) \cdot (-1, 0, 0) = 1 + 0 + 0 = 1 > 0, \text{ so surface is hidden.}$$

- For Fig. 9.3.1(b), the shaded surface is parallel to YZ plane and it's outward normal is in the positive X direction.

$$\text{So, } N = (1, 0, 0) \text{ and } V = (-1, 0, 0)$$

$$N \cdot V = (1, 0, 0) \cdot (-1, 0, 0) = -1 + 0 + 0 = -1 < 0, \text{ so surface is visible.}$$

- Here Normal to the plane and viewing direction are parallel to each other but they are in the opposite direction.

- In Fig. 9.3.1(c), the shaded surface is parallel to the XY plane, and its normal is in the positive Z direction. So the surface is parallel to viewing direction and normal to the surface and viewing direction are perpendicular to each other.

$$\text{So, } N = (0, 0, 1) \text{ and } V = (-1, 0, 0)$$

$$N \cdot V = (0, 0, 1) \cdot (-1, 0, 0) = 0 + 0 + 0$$

$$= 0, \text{ so surface is parallel to viewing direction}$$

- If the angle between viewing direction and normal to the plane is between $+90^\circ$ to -90° , then the surface is visible, otherwise it is hidden and should be removed.

- If viewing direction is along Z-axis then let's say view direction vector is $V = (0, 0, V_z)$. Assume that the normal vector is defined by vector $N = (A, B, C)$.

$$N \cdot V = (A, B, C) \cdot (0, 0, V_z) = V_z \cdot C$$

- If we consider the right-hand viewing system, with viewing direction in the negative Z-axis, then the surface is back faced if $C \leq 0$ and surface is visible if $C > 0$.



- For convex polyhedron like a cube, surfaces are either fully visible or completely hidden.
- But concave polyhedrons like shown in Fig. 9.3.2 may have some surfaces partially visible, for that we need to perform some more tests to decide which surface is visible and what portion of it is visible.

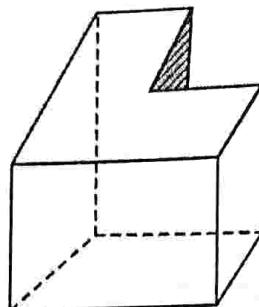


Fig. 9.3.2 : Concave polyhedron

9.4 Depth Buffer Method

Q. Explain the Z-Buffer algorithm for hidden surface removal.

MU - May 18, May 19, 10 Marks

Q. Write a short note on depth buffer method.

MU - Dec. 18, 10 Marks

- **Depth buffer method** is also known as the **z-buffer method**. This is an *image space* approach.
- It is a simple, widely used and efficient method to decide the visibility of the pixel.
- It compares the depth of the pixels of overlapping objects into the scene and renders the nearest pixel to the viewer.
- This algorithm is more suited to polygons, but it can also be extended for objects with non-planar surfaces with additional computation.
- As shown in Fig. 9.4.1, depth of corresponding pixels of overlapping objects are compared and the nearest pixels are projected.
- This algorithm is generally applied on normalized coordinates, where all values fall between 0 and 1.
- This algorithm is an extension of the idea of the frame buffer. Implementation requires two buffers of the size of screen scanned pixel by pixel, the depth of each object is calculated and compared. Depth of the nearest pixel to the viewer is stored in z-buffer and its intensity is stored in i-buffer at corresponding spatial location(x, y).
- If no object exists at a location (x, y) in the scene then z is set to zero and intensity will be set to background color for that pixel.

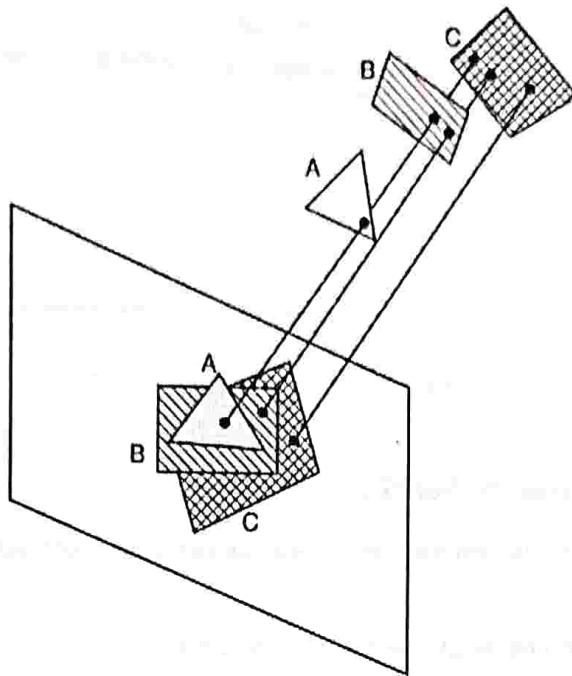


Fig. 9.4.1 : Projection of objects on the plane

Algorithm for depth buffer technique is demonstrated here :

Algorithm DEPTH_BUFFER()

```

//Initialization
for each location (x, y) in buffers do
    Zbuffer (x, y) ← -∞           // depth of farthest plane
    Ibuffer (x, y) ← Ibackground
end

// Rasterization
for each polygon P in scene do
    for each pixel (x, y) in polygon P do
        pz ← Z value of polygon at (x, y)
        if pz > Zbuffer (x, y) then
            Zbuffer (x, y) ← pz
            Ibuffer (x, y) ← Pcolor
        end
    end
end

```



For any plane, z can be calculated as follows :

- Equation of plane is $Ax + By + Cz + D = 0$. Where (x, y, z) is pixel on plane and A, B, C, D are constants describing the plane.

$$\therefore z_{\text{old}} = \frac{-Ax - By - D}{C}$$

- On scan line $Y = y$, next pixel would be $(x + 1, y)$. The depth at a pixel $(x + 1, y)$ is given as,

$$z_{\text{new}} = \frac{-A(x + 1) - By - D}{C} = \frac{-Ax - By - D - A}{C}$$

$$= \frac{-Ax - Bx - D}{C} - \frac{A}{C}$$

$$z_{\text{new}} = z_{\text{old}} - \frac{A}{C}$$

- A and C are constant so A/C ratio is also a constant, we need only one subtraction to find z at adjacent pixel $(x + 1, y)$ on the same scan line.
- Similarly, z for the point (x, y) at next scan line $Y = y + 1$ would be,

$$z_{\text{new}} = \frac{-Ax - B(y + 1) - D}{C} = \frac{-Ax - Bx - D}{C} - \frac{B}{C} = z_{\text{old}} - \frac{B}{C}$$

Advantages

1. Simple to understand.
2. Easy to implement
3. No restriction on the number of polygons in the scene
4. Work for dynamic scene
5. Sorting of objects is not required
6. Fast with hardware support.

Disadvantages

1. Require more memory.
2. Time-consuming.

$$\text{Time} = O(n \cdot d^2)$$

Where d^2 is the size of the scene, and n is a number of objects.

3. Prone to aliasing effect due to rasterization.
4. If we alter the order of polygon for the processing, then also algorithm ends up with the same result.

9.5 Area Subdivision Method

Q. Write a short note on area subdivision method.

MU - Dec. 18, Dec. 19, 10 Marks

This is a divide and conquer approach. It is also known as **area subdivision method**. It utilizes the tendency of human eye-brain information processing. Human brain spends less time on the area with little information, and more time and efforts are spent on an area containing more information. Warnock's algorithm takes advantage of *area coherence*.

Warnock's algorithm evaluates the relationship between the window (area) in the image (object). If the window is not empty, a subdivision of the window is carried out until predefined level of division is achieved or window contains information simple enough to resolve.

There are four possible relationships that a surface can have with an area of subdivided view plane. These relationships are described in Fig. 9.5.1.

- **Surrounding polygons** are completely containing the area (White square) of interest (Fig. 9.5.1 (a))

- **Intersecting polygons** intersect the area (Fig. 9.5.1 (b))

- **Contained polygons** are completely inside the area (Fig. 9.5.1 (c))

- **Disjoint polygons** are completely outside the area (Fig. 9.5.1 (d))

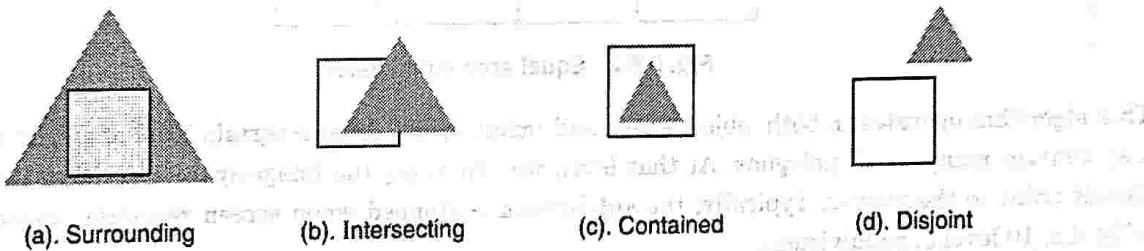


Fig. 9.5.1 : Relationship between two polygons

- If polygons are disjoint (Fig. 9.5.1 (d)), a polygon has no influence on the area of interest. The part of the intersecting polygon which is outside the area is also irrelevant. The part of intersecting polygon which is inside the area can be treated the same as the contained polygon.

- Surface visibility can be decided by the four relationships illustrated in the Fig. 9.5.1. If any of the following cases is true, no further subdivision of area is needed

Case 1 : If all the polygons are disjoint, fill the area with background-colour

Case 2 : If there is only one container or intersecting polygon, then the area is first filled with background color followed by the part of the polygon contained in the area.

Case 3 : If there is only one surrounding polygon then fill the area with polygon color.

Case 4 : If there are more than one intersecting, overlapping or inside polygons, more processing is required.

- Case 4 is further described in Fig. 9.5.2.

- In Fig. 9.5.2, all intersection point of surrounding polygon is near to viewer than the rest of all intersection points of intersecting and contained polygon. So the area of interest is colored with a surrounding polygon.



- However, in Fig. 9.5.2, surrounding polygon is in front of remaining polygons but it's all intersection points are not close to the viewer. Depth sort algorithm accepts this case, but Warnock's algorithm performs subdivision until one of the four relations holds or termination criteria are matched. After performing subdivision, no need to examine disjoint (case 1), intersecting (case 2) and surrounding (case 3) polygons. Only those cells are subdivided which contains multiple polygons (case 4).

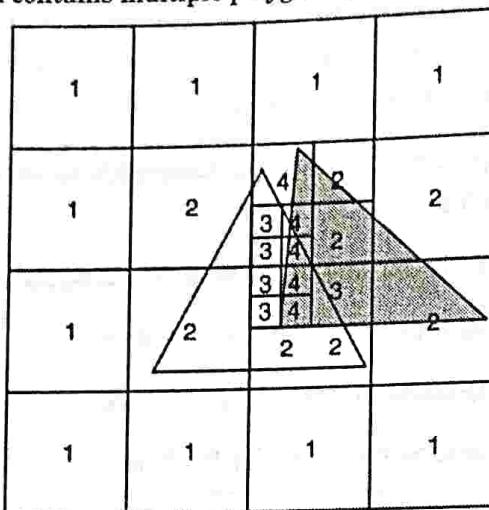


Fig. 9.5.2 : Equal area subdivision

- This algorithm operates in both, object space and image space. After a certain level of subdivision, the scene may contain many small polygons. At that level, we can apply the image-space algorithm to determine the closest point to the viewer. Typically, the subdivision is stopped when screen resolution reaches to 1024 by 1024 (i.e. 10 level of subdivision).
- After that, the depth of the individual area is computed and the area is displayed with the polygon color which is nearest to the user. Antialiasing methods are applied to further improve the display quality.
- Equal area subdivision is shown in Fig. 9.5.3 creates unnecessary divisions. As an alternative to that, we can use division about vertex as shown in Fig. 9.5.3. In this approach, subdivision takes place at vertex only. In the mentioned diagram, First subdivision is carried out at point P followed by second division at point Q.

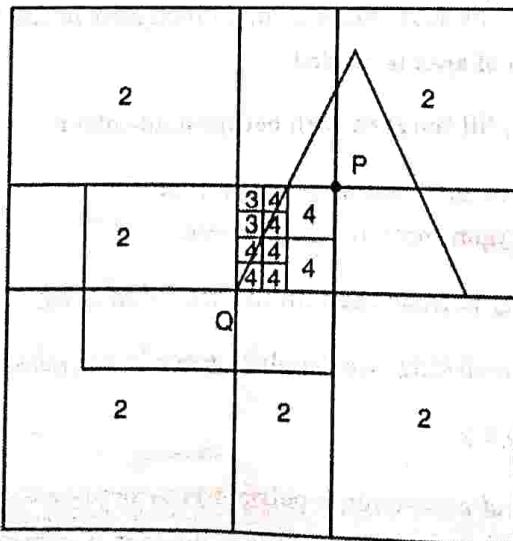


Fig. 9.5.3 : Subdivision from vertex

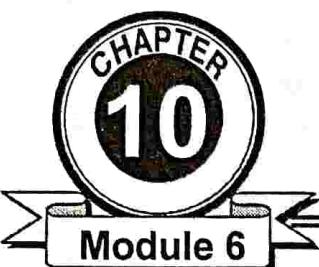
Applications

- Used for hidden surface removal.
- Can make rendering possible for the complicated image by subdividing it.
- Quickly process the area with little information.

Review Questions

- Q. 1 Explain visible surface detection.
- Q. 2 Write a note on object space and image space approach.
- Q. 3 Explain Back face detection and removal.
- Q. 4 Describe the Z-buffer hidden surface algorithm.
- Q. 5 Explain Warnock's hidden face removal algorithm.

□□□



Animation

Syllabus

Introduction to Animation, Traditional Animation Techniques, Principles of Animation, Key framing : Character and Facial Animation, Deformation, Motion capture

10.1 Introduction

- Computer animation refers to the time sequence of visual changes. The animation is a key concept in entertainment, education, simulation, games, scientific and engineering studies etc. Today, the scope of animation is not only limited to changing the position of an object with time, but the concept is extended to various transformation operations like scaling, rotation, variations in color, transparency, surface property, shape etc.
- The animation is achieved by displaying successive frames with minor differences. Near identical frames are displayed at a certain rate, which creates the illusion of animation in the scene.
- Entertainment and advertisement animation often transform one shape to others, like a man is converted to animal or car. Computer animation can be generated by changing camera position, orientation, movement speed, focal length, light, etc.
- Accuracy is a concern when it is to be used in study and simulation. Entertainment and advertisement applications are more concerned about visual and lighting effects.

10.2 Traditional Animation Techniques

- The animation is widely used in movies, advertisement, games etc. Animation gives life to the objects present in the scene through various transformation operations. Animation can be computer-assisted or computer-generated.
- The idea behind the animation is to fool the human eye by displaying images at a certain speed so that they are interpreted as video of moving objects.
- Animation techniques can be classified as conventional animation and computer-based animation.

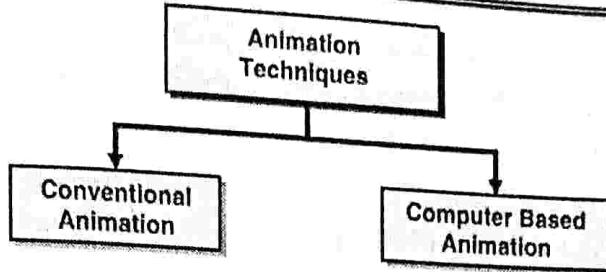


Fig. 10.2.1 : Types of animation techniques

- In the following subsections, we will discuss both the techniques
- In the conventional approach, all the frames of video are designed by hand. Frames are displayed at the rate of least at 24 frames per second. The conventional method takes a tremendous amount of time and effort to create a video.
- Conventional animation is carried out in the following steps :
 - o **Storyboard** : The sequence of animation is first drafted on paper in the form of sketches, which is called a storyboard.
 - o **Keyframes** : Keyframes are the first frame of every shot in the sequence. Keyframes are used to interpolate the animation sequence. As the animation is achieved using keyframes, such type of animation is also known as **key-frame animation**.
 - o **Inbetweens** : Inbetweens are the set of frames between successive keyframes. Objects in motion gradually move in subsequent frames and reach to the destination in next keyframes.
 - o **Pencil Test** : Before the finalizing the complete animation sequence, trial video is created from keyframes and inbetweens, which is called a pencil test.
 - o **Cels** : Instead of applying animation to all the objects together, the animation scene is decomposed into small parts called cels. These cels may be independent or partially dependent. Each cel is animated independently. This cel representation adds more flexibility to the animation.
 - o **Route Sheet** : Route sheet is one of the best methods to know exactly how each step of the project has progressed towards completion of the project.
 - o **Exposure sheet** : This is also known as camera instruction sheet or x-sheet. This tool allows the animator to organize his thoughts and give instruction to the camera operator how to shoot the animation. This sheet contains the minute description of each frame, including the position of the object, the position of the camera, dialogue, background etc.
- Conventional animation method is tedious but it provides great control over the animation to be created. Conventional animation is not limited by available computing technology. For high-quality animation, it is still faster than computer-based animation. The animation is only limited by the ability of the artist.

10.3 Principles of Animation

Animators at Disney have invented 12 principles for the animation. These principles not only add the knowledge on how to achieve the animation but they also help to create appealing more alive animation. All principles are discussed here.



- **Squash and stretch :** Squash and stretch define the elasticity, volume and flexibility of the character. Squash and stretch are used for all kind of objects in the scene. It also used in facial animation. It is used in almost every type of animation like a bouncing ball to moving person.
- **Anticipation :** Anticipation prepares the viewer for the main action to come, for example starting to run, jump, kick etc. The first step of anticipation is to squat. Repetitive use of anticipation can be used to achieve the comic effect. All motion contains a lesser or greater amount of anticipation to add a more realistic effect.
- **Staging :** It reflects the use of stage elements, pose, camera motion and action. These elements are essential to demonstrate the attitude, temper and reaction of the character. Use of various shots including long shot, medium shot and close-up helps to narrate the story to the point. Due to time limitation in the video, the story should be conveyed through a clear cut agenda without any repetition. Foreground and background should complement each other to effectively convey the story.
- **Straight ahead action and pose to pose :** Straight head animation start with the first picture and gradually translates to the final scene. Generally, it is used with hand-drawn animation to create a quick sequence of animation. It is difficult to get control of the process with this method.

Pose to pose method is a more organized way of adding more interest in animation. This method gives better control to resize, reshape and rearrange object in the scene.

Many times, both approaches are used simultaneously to complement each other.

- **Follow through and overlapping actions :** While we are travelling in a bus with almost constant, over the body is steady. Our body is in motion with the bus, but it seems steady to insiders. On sudden break, the bus will stop but over the body will move in the forward direction. Similarly, when the bus starts with a jerk, our body goes in a backward direction. This is called follow-through and overlapping. Objects in motion must be supplemented with such actions. For example, when a girl is running, her hair, the dress will be flying in the opposite direction of motion and will remain in the motion for some time even after she stops.
- **Slow-in and Slow-out :** Motion is dynamic. When we start the car, while we drive the car or when we stop the car, speed is different. When object moving, its speed increases gradually, then it might travel with different velocity and at last, when it stops, speed decreases gradually and it becomes stable. This principle is also applied to objects in the animation.
- **Arches :** In the real world, the motion of an object is not always straight forward. Consider the fan, pendulum, ball, hands, legs and many other objects, the circular or arc motion is very natural for the objects. So it is very essential to simulate arc motion in objects to add more realism.
- **Secondary Action :** Secondary actions are the actions which complement primary actions and distract or direct the attention of the viewer. For example, reading a newspaper is the primary action. Same time a person may be smoking or having a cup of tea is the secondary action. Sometimes secondary action becomes primary action when the primary action gets over and secondary option action is still in progress.
- **Timing :** Timing represents how many frames are being used to perform certain actions. To represent sharp and quick action, fewer frames are used. To convey smooth and slower action more frames are used.

Like speed, one can also control weight, size temper etc. with the help of timing. The time required to climb the tree will not be the same for aunt and monkey. Time will be different for car and truck to take U-turn.

- Exaggeration :** There is no limit on the level of animation achieved. Sometimes exaggerated animation help us to understand the concept in a better way. This characteristic is often used in a fight sequence in movies. It adds more dynamics and adds greater expression in the scene.
- Solid drawing and solid posing :** The character pose should be balanced and it should have evenly distributed weight and center of gravity. Poses of character should be expressive. It should unambiguously communicate thoughts, condition, temper feeling etc.
- Appeal :** Let it be hero or villain, aunt or elephant, all character must be charming and appealing irrespective of its role and size in the animation. Many times, negative characters also leave behind a greater impression on the audience. Small characters like Tom, Jerry, Mickey Mouse, Doraemon etc. are so appealing that they have left a great impression on kids and adults.

10.4 Keyframing : Character and Facial Animation

- Keyframe animation does interpolation to generate intermediate frames from a given pair of frames. The animator can control several intermediate frames to be generated. More keyframe generates smooth animation.
- Only a few seconds of animation of a bouncing ball produces many frames. To generate smooth and realistic animation, we should be careful in specifying position and number of keyframes in real sequence. Just by specifying the first and last frame as keyframes, we may end up with generating only liner motion of the ball.
- Following a sequence of frames are keyframes of the bouncing ball motion. These six frames are sufficient to explain and generate an animation of bouncing ball effect.

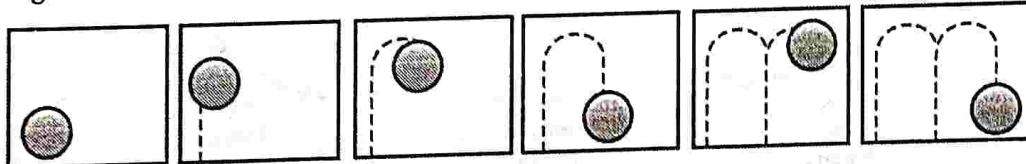


Fig. 10.4.1 : Keyframes for bouncing ball event

- Keyframe animation can be applied to motion, shape, color etc. Motion animation generates the intermediate frames defining the motion of an object at any given moment. In shape transformation, intermediate frame describes the transformation of one shape to another shape. In color transformation, in between frames determines the color change from source to the target frame.

Approaches for keyframe animation

There are two approaches to keyframe animation.

- Shape interpolation :** This approach derives in-between frames by interpolating the shape information. This has a major role in the film industry and advertisement. It performs geometric transformation to change the shape of the object. Motion can be distorted because of interpolation. Fig. 10.4.2 describes the shape interpolation.

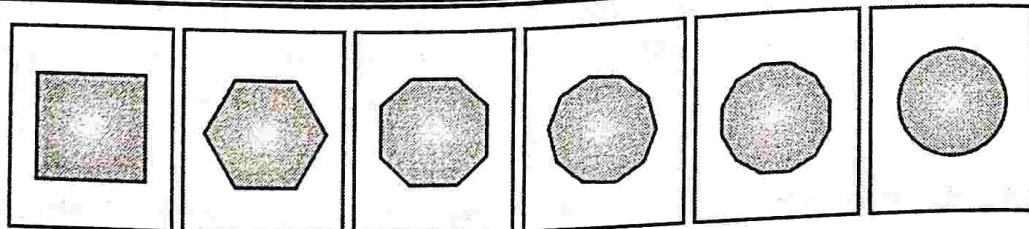


Fig. 10.4.2 : Shape interpolation

The problem arises when the numbers of vertices are not same in both objects. Typically, the pre-processing step is performed to equalize the number of vertices in both objects.

2. **Parameter interpolation :** Shape interpolation is faster but less accurate. Another way is to interpolate the parameters of both objects rather than object shape itself. This is also known as *parametric keyframe animation* or *key transformation animation*. Parameter interpolation considers various types of parameters like visual parameters, physical parameters, spatial parameters etc.

In Fig. 10.4.3, points on keyframe k and $k + 1$ indicates parameters of both frames, intermediate frames gradually map parameters of keyframe 1 to parameters of keyframe $k + 1$.

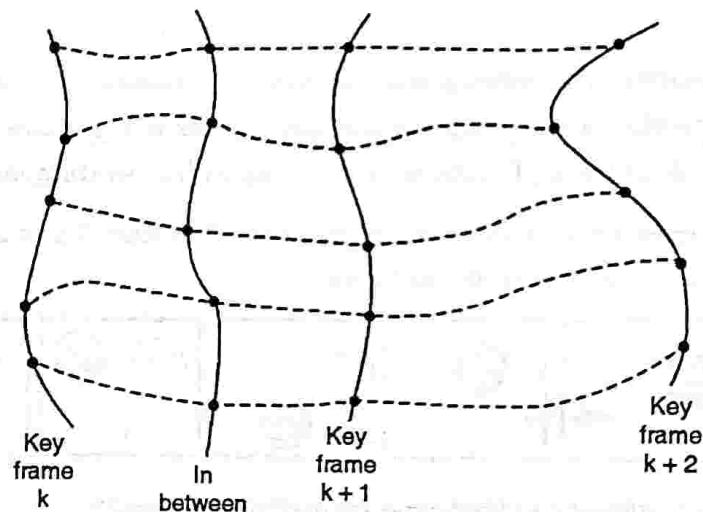


Fig. 10.4.3 : Parameter interpolation

In both approaches, linear interpolation does not give acceptable results. Linear interpolation creates undesired effects like motion discontinuity, lack of smoothness in motion, distortion in rotation etc. Spline interpolation like quadratic or cubic interpolation can get rid of all these unwanted effects of linear interpolation.

10.5 Deformation

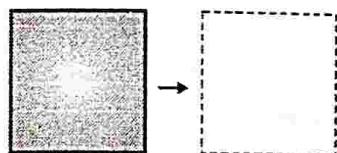
- Every object, light, material etc. used in animation is defined by the set of parameters. The animation is achieved by changing such parameters over time. Animation in main characters/objects is the prime concern. Animation in the rest of the characters/objects supplements the animation. Objects are animated using motion and deformation.

The process of animation can be setup by the following steps :

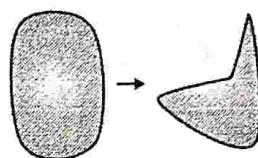
1. **Specifying deformation** : This is achieved by deciding the hierarchy of animation, skinning, blending of shapes etc.

2. **Providing animation control** : This is achieved by specifying pivot or control points and by creating dummy shapes to achieve the sequence of animation.

- Deformation is achieved by either repositioning the object or by reshaping the object. The first one is called rigid body transformation, in which the shape of the object does not alter. The second one is called deformation in which the shape of the object is deformed. Fig. 10.5.1 illustrates both the concepts.



(a) Rigid body transformation



(b) Shape deformation

Fig. 10.5.1 : Deformation types

10.5.1 Rigid Body Transformation

- In rigid body transformation, the object moves along straight, circular or any random path. The shape and size of the object will remain intact after the transformation. Translation, rotation and reflection are examples of rigid body transformation.
- Even after performing rigid body transformation, the relative position of the points in an object does not change.
- Though it is simple, it does not simulate the real objects. Real objects frequently change the shape, size and orientation. Rigid body transformation only provides mathematical abstraction.
- Various rigid body transformations are described in the Fig. 10.5.2 :

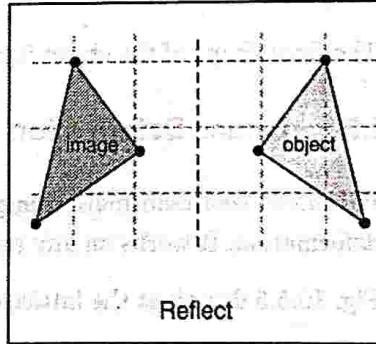
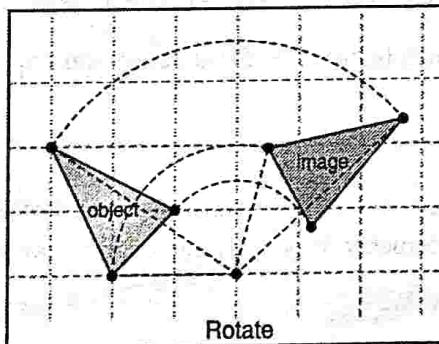
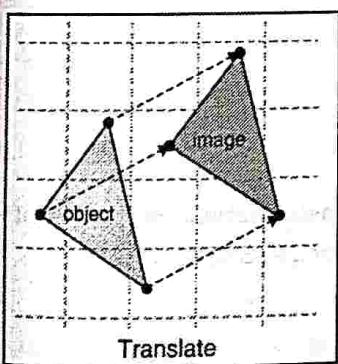


Fig. 10.5.2 : Rigid body transformations

- All of these functions are linear operations and can be effectively represented by matrix multiplication.

- The transformed coordinates of points P is computed as,

$$P' = M \cdot P$$

Where M represents the corresponding transformation matrix.



10.5.2 Deformation

- In deformation, the shape of the object does change and the relative position of object points also change. It is useful in achieving more realistic animation. Transformation is good for hard objects, whereas animation in a soft object is effectively represented by deformation.
- Deformation affects all points independently. With this, we can achieve any kind of animation due to freedom of shape deformation to any degree. The object deformation is shown in Fig. 10.5.3 :

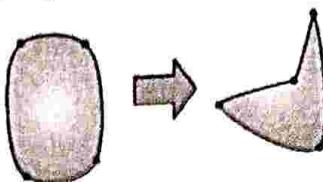


Fig. 10.5.3 : Deformation

- Deformation can be achieved using different ways.

10.5.2.1 Parameterized Deformation

- It defines the deformation function over control points. The same set of control points can be deformed in different ways by changing function and parameter. The deformation can be defined as :

$$P' = f(P, \{\alpha_i\})$$

- Where P is the set of object points and f is the deformation function with parameters α_i . Original objects (solid line) and deformed objects (dashed line) are shown in the Fig. 10.5.4.

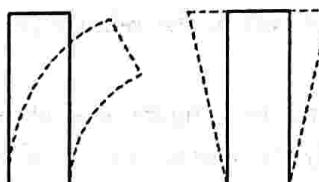


Fig. 10.5.4 : Original and deformed objects

- The final shape of the object is determined by function f and parameters α_i .

10.5.2.2 Lattice Deformation

- Parameterized deformation might change the object shape abruptly. Lattice transformation creates smooth deformation. It works on any random geometry. It is also simpler than parameterized animation.
- Fig. 10.5.5 describes the lattice deformation.

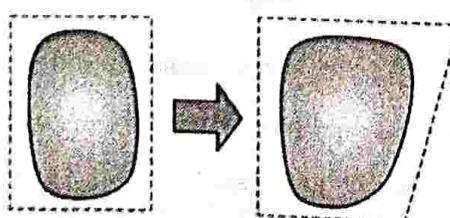


Fig. 10.5.5 : Lattice deformation

10.5.2.3 Composite Deformation

- Complex deformation is achieved by function composition. Multiple functions are applied in sequence. The process is described as follow :

$$P' = f_1(f_2(P))$$

10.6 Motion Capture

- Motion capture character animation is the process of recording an actor's movement and mapping it to 3D characters in the computer.
- To achieve this, sensors are attached all over the body of the actor. Movement of sensors is tracked and recorded and mapped on the 3D model in computer in real-time. After retakes, appropriate take is selected and the motion is applied to the 3D character.
- Nowadays, motion capture is widely used in film and animation industries. Instead of generating individual frames from keyframes in the computer, the entire sequence is derived by tracing the movement of sensors.
- Even when animators create a movement of animation characters by hand, they refer to video footage, study action on screen, even they look themselves in the mirror. Creating digital animation by hand is known as "keyframing".
- Fig. 10.6.1 shows the use of motion capture in a very well-known Hollywood movie "AVATAR".



Fig. 10.6.1 : Motion capture in movie "AVATAR"

- Apart from film industries and games, sport and athletics do a lot of work with motion capture technology. For training the racer, sensors are attached to bike and rider. The motion of these sensors is recorded and analyzed. The racer will be instructed to adjust his posture accordingly in real-time through web data received from sensors.

Review Questions

- Q. 1 What is animation?
- Q. 2 Discuss the steps of design of animation sequences.
- Q. 3 Discuss various types of animation language.
- Q. 4 Write a short note on a keyframe.
- Q. 5 What is the importance of keyframe? How it is useful in animation?



- Q. 6 Explain the use of motion specification in animation.
- Q. 7 Write a short note on motion specification methods based on :
- (i) Geometric and kinematics information.
 - (ii) Specification methods based on physical information.
 - (iii) Motion specification methods based on behavioural information.

Model Question Paper

Model Question Paper

Computer Graphics (CSC305)

Semester - III (Computer Engineering)

Time : 3 Hour

Maximum Marks : 80

Instruction to the candidates :

- 1) Question No.1 is compulsory**
- 2) Attempt any three of remaining five questions**
- 3) Assume any suitable data if necessary and justify the same**

- Q. 1** (a) State and explain any five applications of computer graphics. [5 Marks]
(b) Define the terms : Computer Graphics, Scan conversion, pixel, resolution, aspect ratio, frame buffer [5 Marks]
(c) Write an algorithm for Bresenham's Line drawing method. [5 Marks]
(d) Discuss the characteristics of the ideal line. [5 Marks]
- Q. 2** (a) Explain Bresenham's circle drawing method. [10 Marks]
(b) Explain two methods for testing whether the point is inside the polygon or not. [10 Marks]
- Q. 3** (a) Explain flood fill and boundary fill algorithms. [10 Marks]
(b) Perform 90° rotation of triangle with vertices A (2, 2), B (4, 2) and C (3, 3)
 - 1) About origin
 - 2) About reference point (-2, 2)[10 Marks]
- Q. 4** (a) Derive transformation matrix to perform reflection about a line $y = mx + c$. [10 Marks]
(b) Apply Liang Barsky algorithm to the line 10 with coordinates (35,60) and (80,25) against the window $(X_{\min}, Y_{\min}) = (10,10)$ and $(X_{\max}, Y_{\max}) = (50,50)$. [10 Marks]
- Q. 5** (a) Derive transformation matrix for rotation about an arbitrary axis in space. [10 Marks]
(b) State and discuss principles of animation. [10 Marks]
- Q. 6** (a) Differentiate : Perspective Projection vs. Parallel Projection. [5 Marks]
(b) Describe the continuity in curve. [5 Marks]
(c) Discuss the classification of visible surface detection methods. [5 Marks]
(D) Write a short note on Back Surface Detection Method. [5 Marks]

Note

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

Your Success Is Our Goal
Semester III - Computer Engineering

ENGINEERING MATHEMATICS - III

Dr. Narendrakumar Dasre, Dr. Sachin Wanl, Dr. Pritam Chetan Wani

DISCRETE STRUCTURES AND GRAPH THEORY

Dr. Bhakti Raul-Palkar

DATA STRUCTURE

Dilip Kumar Sultania

DIGITAL LOGIC & COMPUTER ORGANIZATION & ARCHITECTURE

J. S. Katre, Harish G. Narula

COMPUTER GRAPHICS

Dr. Mahesh M. Goyani

eBooks are available on www.techknowledgebooks.com

coming soon.....



es[®]
easy-solutions

Paper Solutions Trusted by lakhs of students from more than 20 years



Head Office :
B/5, First Floor, Maniratna Complex, Taware Colony,
Aranyeshwar Corner, Pune - 411009. Maharashtra, India.
Tel. : 91-20-24221234, 91-20-24225678



ISBN : 978-93-89889-99-4



9 789389 889994

Price ₹ 425/-

M0148C



Distributors

Vidyarthi Sales Agencies
T. : (022) 23867279, 98197 76110

Ved Book Distributors
80975 71421 / 92208 77214 / 80973 75002

Bharat Sales Agency
T. : (022) 23819359, 86572 92797

Our Branches : Pune | Mumbai | Kolhapur | Nagpur | Solapur | Nashik

For Library Orders Contact - Ved Book Distributors M : 80975 71421 / 80973 75002

Email : Info@techknowledgebooks.com | Website : www.techknowledgebooks.com

Like us at:



TechknowledgePublications