NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIALS

CS 401 COMPUTER GRAPHICS

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- Established in: 2002
- Course offered : B.Tech in Computer Science and Engineering

  M.Tech in Computer Science and Engineering

  M.Tech in Cyber Security
- Approved by AICTE New Delhi and Accredited by NAAC
- Affiliated to the University of A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.
5.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.

PEO2: Graduates will be able to analyze, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.

PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.

PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamwork and leadership qualities.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOMES

C401.1	Demonstrate Various Graphics Devices
C401.2	Analyze and implement algorithms for Line, Circle, Polygon drawing
C401.3	Apply geometrical transformation on2D Objects
C401.4	Analyze and implement algorithms for clipping and illustrate 3D graphics representations.
C401.5	Apply various projection technics on 3D objects and hidden line elimination techniques.
C401.6	Demonstrate the various concepts of image processing

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

CO’S	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10		PO11	PO12
C401.1	3	-	2	-	-	-	-	-	-	-		-	3
C401.2	3		3	2	3	-	-	-	-	-		-	-
C401.3	3		3	-	3	-	-	-	-	-		-	2
C401.4	3	3	2	-	2	-	-	-	-	-		-	2
C401.5	3	2	3		3	-	-	-	-	-		-	3
C401.6	3	3	3	3	3	-	-	-	-	-		-	3
C401	3	2.67	2.67	2.5	2.8	-	-	-	-	-		-	2.6

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

CO’S	PSO1	PSO2	PSO3
C401.1	3		
C401.2	3		
C401.3		3	
C401.4		3	
C401.5			3
C401.6	3		
C401	3	3	3

SYLLABUS

Course code	Course Name	L-T-P Credits	Year of Introduction
CS401	COMPUTER GRAPHICS	4-0-0-4	2016

Course Objectives :

- To introduce concepts of graphics input and display devices.
- To discuss line and circle drawing algorithms.
- To introduce 2D and 3D transformations and projections.
- To introduce fundamentals of image processing.

Syllabus:

Basic Concepts in Computer Graphics. Input devices. Display devices. Line and circle drawing Algorithms. Solid area scan-conversion. Polygon filling. Two dimensional transformations. Windowing, clipping. 3D Graphics, 3D transformations. Projections – Parallel, Perspective. Hidden Line Elimination Algorithms. Image processing – digital image representation – edge detection – Robert, Sobel, Canny edge detectors. Scene segmentation and labeling – region-labeling algorithm – perimeter measurement.

Expected Outcome:

The Students will be able to :

i. compare various graphics devices
ii. analyze and implement algorithms for line drawing, circle drawing and polygon filling
iii. apply geometrical transformation on 2D and 3D objects
iv. analyze and implement algorithms for clipping
v. apply various projection techniques on 3D objects
vi. summarize visible surface detection methods
vii. interpret various concepts and basic operations of image processing

Text Books:

1. Donald Hearn and M. Pauline Baker, Computer Graphics, PHI, 2e, 1996
2. E. Gose, R. Johnsonbaugh and S. Jost., Pattern Recognition and Image Analysis, PHI PTR, 1996 (Module VI – Image Processing part)
3. William M. Newman and Robert F. Sproull , Principles of Interactive Computer Graphics. McGraw Hill, 2e, 1979
4. Zhigang Xiang and Roy Plastock, Computer Graphics (Schaum's outline Series), McGraw Hill, 1986.

References:

1. David F. Rogers , Procedural Elements for Computer Graphics, Tata McGraw Hill, 2001.
2. M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis, and Machine Vision, Thomson India Edition, 2007.
3. Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing. Pearson, 2017

Course Plan			
Module	**Contents**	**Hours**	**End Sem. Exam Marks**
I	Basic concepts in Computer Graphics – Types of Graphic Devices – Interactive Graphic inputs – Raster Scan and Random Scan Displays.	7	15%
II	Line Drawing Algorithm- DDA, Bresenham's algorithm – Circle Generation Algorithms –Mid point circle algorithm, Bresenham's algorithm- Scan Conversion-frame buffers – solid area scan conversion – polygon filling algorithms	8	15%
FIRST INTERNAL EXAM			
III	Two dimensional transformations. Homogeneous coordinate systems – matrix formulation and concatenation of transformations. Windowing concepts –Window to Viewport Transformation- Two dimensional clipping-Line clipping – Cohen Sutherland, Midpoint Subdivision algorithm	8	15%
IV	Polygon clipping-Sutherland Hodgeman algorithm, Weiler-Atherton algorithm, Three dimensional object representation- Polygon surfaces, Quadric surfaces – Basic 3D transformations	8	15%
SECOND INTERNAL EXAM			
V	Projections – Parallel and perspective projections – vanishing points. Visible surface detection methods– Back face removal- Z-Buffer algorithm, A-buffer algorithm, Depth-sorting method, Scan line algorithm.	9	20%
VI	Image processing – Introduction - Fundamental steps in image processing – digital image representations – relationship between pixels – gray level histogram –spatial convolution and correlation – edge detection – Robert, Prewitt, Sobel.	8	20%
END SEMESTER EXAM			

Question Paper Pattern (End semester exam)

1. There will be ***FOUR*** parts in the question paper – **A, B, C, D**
2. **Part A**
    a. **Total marks : 40**
    b. ***TEN*** questions, each have **4 marks**, covering **all the SIX modules** (***THREE*** questions from **modules I & II**; ***THREE*** questions from **modules III & IV**; ***FOUR*** questions from **modules V & VI**).
    All the TEN questions have to be answered.
3. **Part B**
    a. **Total marks : 18**
    b. ***THREE*** questions, each having **9 marks**. One question is from **module I**; one question is from **module II**; one question ***uniformly*** covers **modules I & II**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
4. **Part C**
    a. **Total marks : 18**
    b. ***THREE*** questions, each having **9 marks**. One question is from **module III**; one question is from **module IV**; one question ***uniformly*** covers **modules III & IV**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
5. **Part D**
    a. **Total marks : 24**
    b. ***THREE*** questions, each having **12 marks**. One question is from **module V**; one question is from **module VI**; one question ***uniformly*** covers **modules V & VI**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
6. There will be ***AT LEAST* 50%** analytical/numerical questions in all possible combinations of question choices.

QUESTION BANK

CS401-COMPUTER GRAPHICS

MODULE I				
SL NO.	QUESTIONS	COS	KL	PAGE NO.
1.	Explain the working of beam penetration method	CO1	K5	24
2.	Explain the working of random scan system with suitable diagram	CO1	K5	31
3.	Describe the working about shadow mask CRT with suitable diagram	CO1	K2	25
5.	Differentiate between random scan and raster scan system	CO1	K4	36
6.	Describe simple random scan display system	CO1	K2	14
7.	Describe flat panel display and its different categories	CO1	K2	27
8.	Write any two interactive graphic input devices	CO1	K1	15
9.	What do you mean by aspect ratio and resolution of a display screen in a raster scan display	CO1	K1	19
10.	Explain the working of raster scan display	CO1	K5	19
11.	Differentiate emmisive and non emissive displays	CO1	K4	27
12.	Explain the working of LED	CO1	K3	29
MODULE II				
1.	Write the midpoint circle drawing algorithm	CO2	K1	53
2.	Write bresenhams line drawing algorithm and plot the points on the line having given endpoints using bresenhams line drawing algorithm	CO2	K3	45

3.	Describe ODD-EVEN rule test and Non zero winding number	CO2	K2	76
4.	Write flood fill polygon filling algorithm	CO2	K1	81
5.	Describe the relevance and various methods used in inside-outside test used in polygon filling	CO2	K2	76
6.	Scan convert the line segments with given end points using DDA line algorithm	CO2	K5	39
7.	Which are the steps involved scan line polygon filling algorithm	CO2	K5	69
8.	Write flood fill polygon filling algorithm	CO2	K1	81
	MODULE III			
1.	Which are the steps involved in window to viewport coordinate transformation in 2D	CO3	K1	109
2.	Derive an equation for window to viewport transformation	CO3	K5	110
3. .	For the given triangle Find the coordinates of vertices after each of the following transformation i)Reflection about the line x=y ii) Rotation of the triangle ABC about vertex A in clockwise direction for an angle 90 degree	CO3	K5	106
4.	Give the equation and matrix form representation of scaling transformation	CO3	K1	90
5.	Give the matrix representation of translation and rotation in homogenous coordinate	CO3	K1	94
6.	Describe about shear	CO3	K2	102
7.	Explain midpoint algorithm with example	CO3	K3	121
8.	Describe cohen Sutherland line clipping algorithm with example	CO3	K2	115
	MODULE IV			
1.	What are the different tables used for polygon surfaces. Illustrate each with an example.	CO4	K1	134
2.	Briefly explain the basic 3D transformations with its equation	CO4	K5	141

3.	Explain Sutherland Hodgeman polygon clipping algorithm with illustration.	CO4	K5	124
4.	Write the two types of polygon meshes with example	CO4	K1	130
5.	Explain about weiler Atherton algorithm ,its different steps and illustrate the algorithm with an example	CO4	K5	129
6.	Briefly explain about quadric surfaces and write down about sphere and ellipsoid	CO4	K5	132
7.	Explain about the polygon surfaces	CO4	K3	134
8.	Write down the plane equation	CO4	K1	137
9.	Write the equations for basic 3D transformation	CO4	K1	141
10	Explain about the different polygon tables	CO4	K3	135
11.	Describe briefly about polygon clipping	CO4	K5	123
12.	Write equation for torus and ellipsoid	CO4	K1	153
	MODULE V			
1.	Write the A -buffer algorithm for hidden surface removal	CO5	K2	185
2.	Differentiate between parallel and perspective projection	CO5	K4	157
3.	Differentiate between image space and object space method	CO5	K4	177
4.	Describe briefly about parallel projection	CO5	K5	158
5.	What are the different categories of axonometric projection	CO5	K2	162
6.	Explain in detail about Z buffer algorithm	CO5	K5	180
7.	Briefly describe about the various visible surface detection algorithms	CO5	K5	176
8.	Differentiate cavalier and cabinet projection	CO5	K4	163
9.	Explain in detail about scan line algorithm for VSD by pointing out the data structure used in this algorithm	CO5	K5	187
10.	Differentiate between orthographic and oblique projection	CO5	K4	160
11.	Explain back face removal algorithm	CO5	K5	178
12.	Describe depth sorting algorithm	CO5	K5	190
	MODULE VI			
1.	What do you understand by the following terms with respect to pixels Neighbours and Adjacency	C06	K1	206
2.	Explain the fundamental steps in digital image processing	C06	K5	195
3.	Briefly explain about prewitt and sobel edge detection method	C06	K5	211
4.	What is edge detection? Explain any one edge detection method in DIP	C06	K1	209
5.	Describe the different components used in DIP	C06	K2	199
6.	Explain about bitmap image	C06	K3	202

APPENDIX 1		
CONTENT BEYOND THE SYLLABUS		
S:NO;	**TOPIC**	**PAGE NO:**
1.	Types of Curves	214
2.	Visual Perception	217

MODULE NOTES

.

.

.

.

.

.

MOD - I

Basic Concepts in Computer Graphics

Computer Graphics is a process of creation, manipulation, storage and display of pictures and experimental data for proper visualization using a computer.

Computer graphics system comprises of a host computer with support of fast processor, large memory, frame buffer and Graphics devices.

Applications of Computer Graphics

1. Computer Aided Design (CAD)

A major use of computer graphics is in design processes, CAD methods are used in the design of buildings, automobile, aircraft, watercraft, spacecraft, computers, textiles and many other products.

2. presentation Graphics

presentation Graphics is used to produce illustrations for Reports or to generate slides or transparencies for use with projectors. It is commonly used to summerize financial, statistical, mathematical, scientific and economic data for Research Reports, managerial Reports, Consumer information bulletins and other types of Reports.

3. Computer Art

Computer graphics methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods, including special purpose hardware, artists' paintbrush programs (such as Lumena), other paint packages. Specially developed software, symbolic mathematics packages. CAD packages, desktop publishing software and animation packages that provide facilities for designing object shapes and specifying object motions.

4. Entertainment

Computer graphics methods are commonly used in making motion pictures, music videos and television shows. Sometimes the graphics scenes are displayed by themselves and sometimes graphics objects are combined with the actors and live scenes. Graphics is used in morphing

5. Education and Training

Computer generated models of physical, financial and economic systems are often used as educational aids.
For some training applications special systems like simulators are designed for the practical session or training of ship captains, aircraft piolets, heavy-equipment operators and air-traffic-control.

6. Visualization

Scientists, engineers, medical personnel, business analysts and others often need to analyze large amount of information or the study behaviour of certain processes. Numerical simulation carried out on supercomputers frequently produce data files which contains thousands and even millions of data values. Producing graphical representations for scientific engineering and medical data sets and processes is generally referred to as scientific visualization. Business visualization is used in connection with data sets related to commerce, industry and other nonscientific areas.

A collection of 2D or 3D dataset contain scalar values, vectors, higher-order tensors or any combination of these data types. Additional techniques like contour plots, graphs, charts are used to produce data visualizations

Mathematicians, physical scientists and others use visual techniques to analyze mathematical functions and processes.

7. Image processing

Image processing is a technique to interpret or modify the existing pictures such as photographs and TV scans. The two application of image processions are improving picture quality and machine perception of visual information used in robotics.

Types of Graphics Devices

Graphics systems comprises of a host computer with support of fast processor, large m/m, frame buffer and

- Display devices
- Input devices
- Output devices
- Interfacing devices

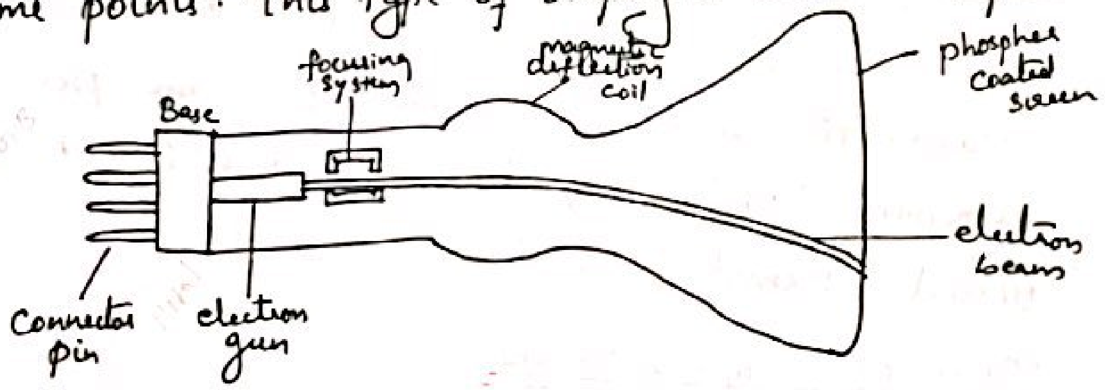
The primary output devices in a graphics system is a video monitor. The operation of most video monitors is based

Scanned with CamScanner

on the Standard Cathode-Ray tube (CRT)

Cathode-Ray Tube

The following figure illustrate the basic operation of a CRT. A beam of electrons (cathode rays) emitted by an electron gun passes through focusing and deflection systems that direct the beam towards specified positions on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. The light emitted by the phosphor fades very rapidly and to keep the phosphor glowing, the picture is redrawn repeatedly by quickly directing the electron beam back over the same points. This type of display is called a Refresh CRT



The primary components of an electron gun in a CRT are the heated metal cathode and a control grid. Heat is supplied to the cathode by directing a current through a coil of wire called filament, inside the cylindrical cathode structure. This causes electrons to be boiled off the hot cathode surface. In the vacuum inside the CRT envelope, the free negatively charged electrons are then accelerated towards the phosphor coating by a high positive voltage. The accelerating voltage can be generated with a positively charged metal coating on the inside of the CRT envelope near the phosphor screen or an accelerating anode can be used.

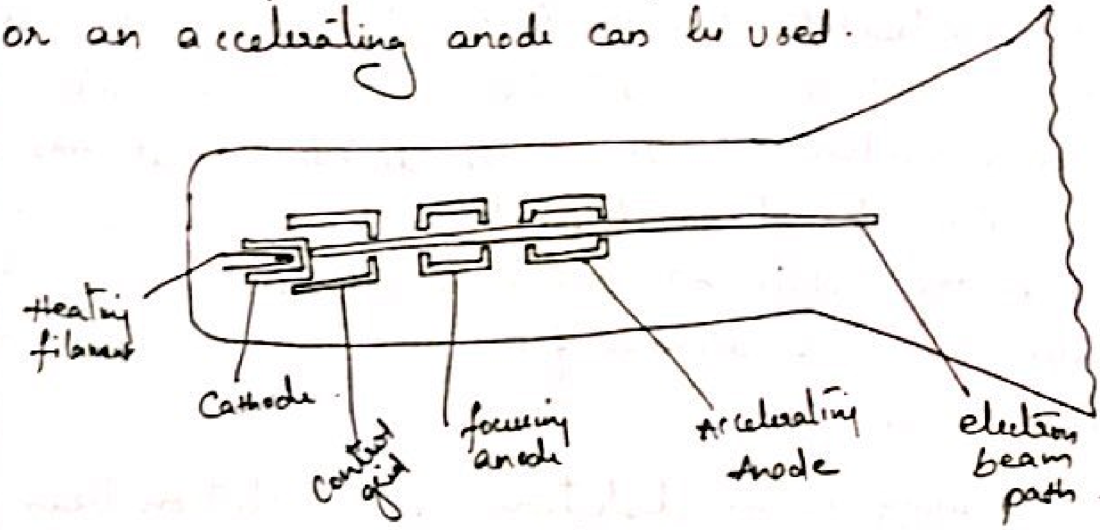


fig: components of CRT.

Intensity of the electron beam is controlling by setting the voltage levels on the control grid, which is a metal cylinder that fits over the cathode. A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stopping them from ~~the~~ passing through the small hole at the end of the control grid structure. A small negative voltage on the control grid simply decrease the number of electrons passing through.

The amount of light emitted by the phosphor coating depend on the number of electrons striking the screen. The brightness of the display can be controlled by varying the voltage on the control grid.

The focusing system in the CRT is needed to force the electron beam to converge into a small spot as it strikes the phosphor. Otherwise the electrons would repel each other,

Scanned with CamScanner

and the beam would spread out as it approaches the screen. Focusing is done with either electric or magnetic field. Electrostatic focusing is mainly used in television and computer graphics monitors. for high precision systems additional focusing hardware is used. The distance that the electron beam must travel to different points on the screen varies because the radius of curvature for most of the CRT's is greater than the distance from the focusing system to the screen centre

With focusing system, deflection of the electron beam can be controlled either with electric fields or with magnetic fields. Cathode-ray tubes are commonly constructed with magnetic deflection coils mounted on the outside of the CRT envelop as shown in the above fig. Two pairs of coils are used, with the coils in each pair mounted on opposite sides of the neck of the CRT envelope. one pair is mounted on the top and bottom of the neck, and the other pair is mounted on opposite sides of the neck. The magnetic field produced by each pair of coils results in a transverse deflection force that is perpendicular both to the direction of the magnetic field and to the direction of travel of the electron beam. Horizontal and vertical deflection is accomplished by two different pairs of coils.

When electrostatic deflection is used, two pairs of parallel plates are mounted inside the CRT envelope. one pair of plates is mounted horizontally to control the vertical deflection, and other pair is mounted vertically to control horizontal deflection.

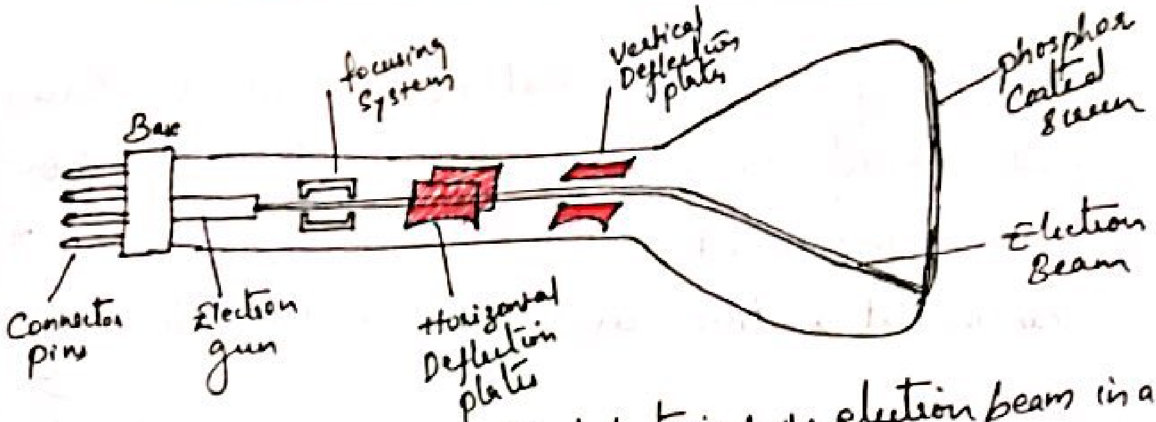


fig : Electrostatic deflection of the electron beam in a CRT

<u>Resolution</u> is the maximum number of points that can be displayed without overlap on a CRT. Resolution of a CRT is dependent on the type of phosphor, intensity to be displayed, focusing system and deflection systems. Higher Resolution available on many systems is 1280 and 1024. They are often referred to as <u>high-definition system</u>.

<u>Aspect ratio</u> is another property of video monitors. This number gives the ratio of vertical points to horizontal points necessary to produce equal-length lines in both directions on the screen.

Raster Scan Display

The most common type of graphics monitor employing a CRT is the raster-scan display, based on television technology. In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

picture definition is stored in a memory area called <u>refresh buffer</u> or <u>frame buffer</u>. This memory area holds the set of intensity values for all the screen points. the stored intensity values are then retrieved from the refresh buffer and painted on the screen one row (scan line) at a time. as shown in the below figure.

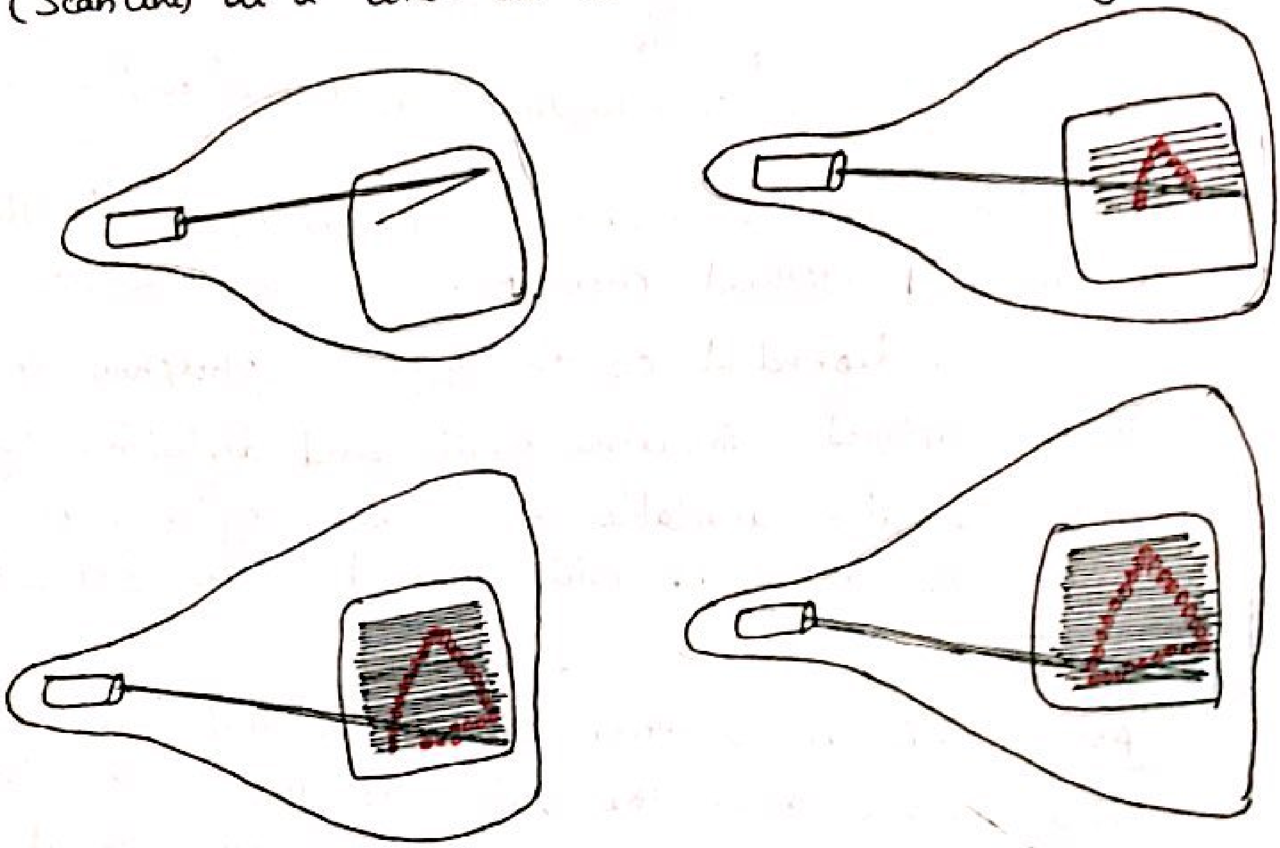


fig: Raster Scan system displays an object as a set of discrete points across each scan line

Each screen point is referred to as a pixel or pel. the capability of a raster scan system to store intensity information for each screen point makes it suited for the realistic display of scenes containing shading and color patterns

Eg: Home television sets and printers are using raster scan methods.

Intensity range for pixel positions depends on the capability of the raster system.

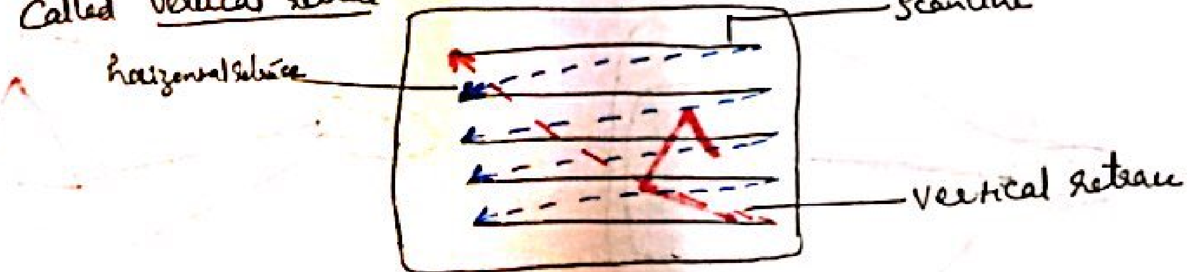
In a Black and white System, each screen point is either on or off. So one bit is needed to control the intensity of screen positions.

For a bitlevel system, a bit value 1 indicates that the electron beam is to be turned on at that position and a value of 0 indicates that the beam intensity is to be off. Additional bits are needed when color and intensity variations can be displayed.

In high quality system upto 24 bits per pixels are used which requires megabytes of storage for the frame buffer.
A system with 24 bits per pixel and a screen resolution of 1024 x 1024 requires 3 mb of storage for frame buffer.
On a black and white systems with one bit per pixel, the frame buffer is commonly called a bitmap and for the system with multiple bits per pixel, the frame buffer is often referred to as a pixmap

Refreshing on raster-scan displays is at the rate of 60 to 80 frames per second.

The return to the left of the screen, after refreshing each scan line is called the horizontal retrace of the electron beam. At the end of each frame, the electron beam returns to the top left corner of the screen to begin the next frame called vertical retrace



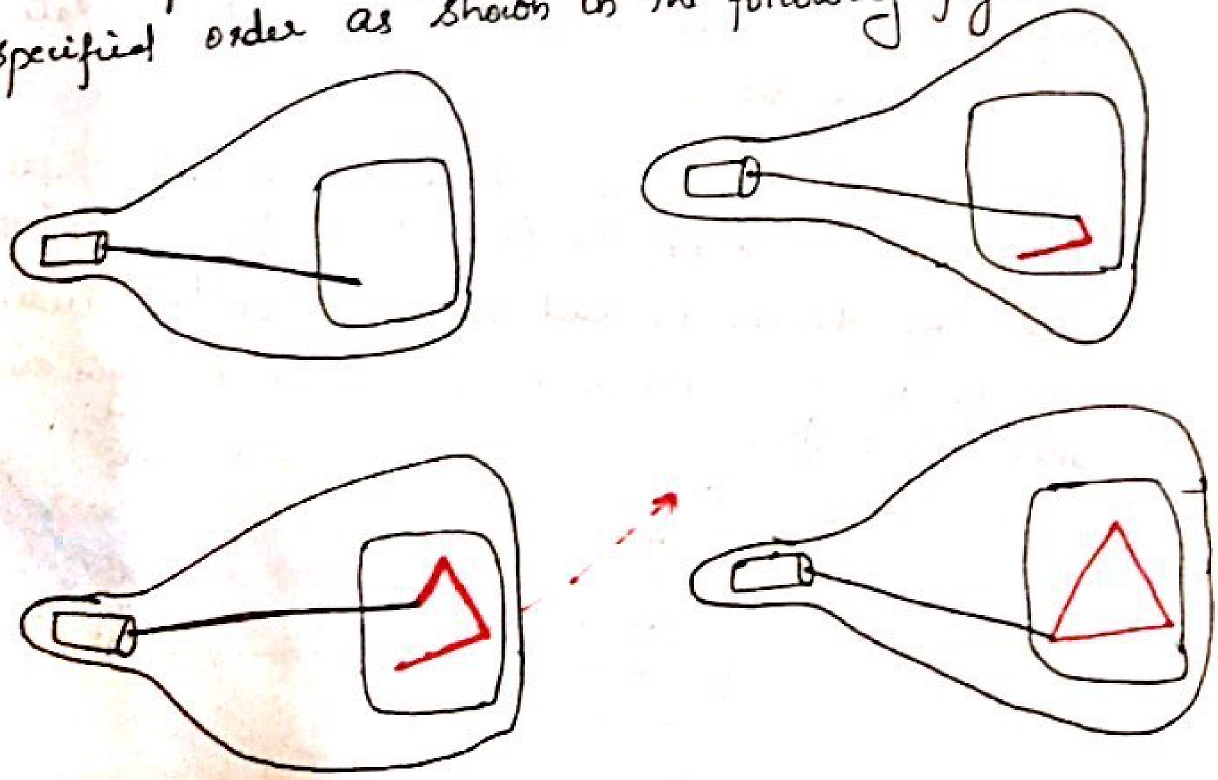
On some raster scan system interlaced refresh procedures are used to display the frames. In the first pass the beam sweeps out the odd numbered scan line from top to bottom. In the second pass all the even numbered scan lines are sweeps across the screen. This method is using with slower refreshing rates.

Random Scan Displays

In random scan display, a CRT has the electron beam directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom.

Random scan monitors draw a picture one line at a time and for this reason random scan are also referred as vector displays or (~~stro~~ stroke writing or Calligraphic display)

The component lines of a picture can be drawn and refreshed by a random scan system in any specified order as shown in the following figure.



Refresh Rate on a Random scan systems depends on the number of lines to be displayed. picture definition is stored as a set of line-drawing commands in an area of memory Refered to as the Refresh display file. Refresh display file is called the display list, display program or simply the Refresh buffer.

To display a specified picture, the systems cycle through the set of commands in the display file, drawing each Component line in turn. After all line drawing commands have been processed, the systems cycles back to the first line command in the list. Random scan displays are ~~disp~~ designed to draw all the component lines of a picture 30 to 60 times each second.

Random-scan systems are designed for line drawing applications and cannot display realistic shaded scenes. Vector Display or Random displays have higher Resolution than raster display because picture definition is stored as a set of line drawing instructions and not as a set of intensity values. for all ~~screen~~ screen points. Vector displays produce smooth line drawings because the CRT beam directly follows the line path. But the Raster system in Contrast produces jagged lines that are plotted as discrete point sets.

Color CRT Monitors

A CRT monitor displays color pictures by using a Combination of phosphors that emit different Colored light. By combining the emitted light from the different phosphors a Range of colors can be generated. Two basic techniques for producing color display with a CRT are the beam penetration and the Shadow mask method

Beam penetration Method:-

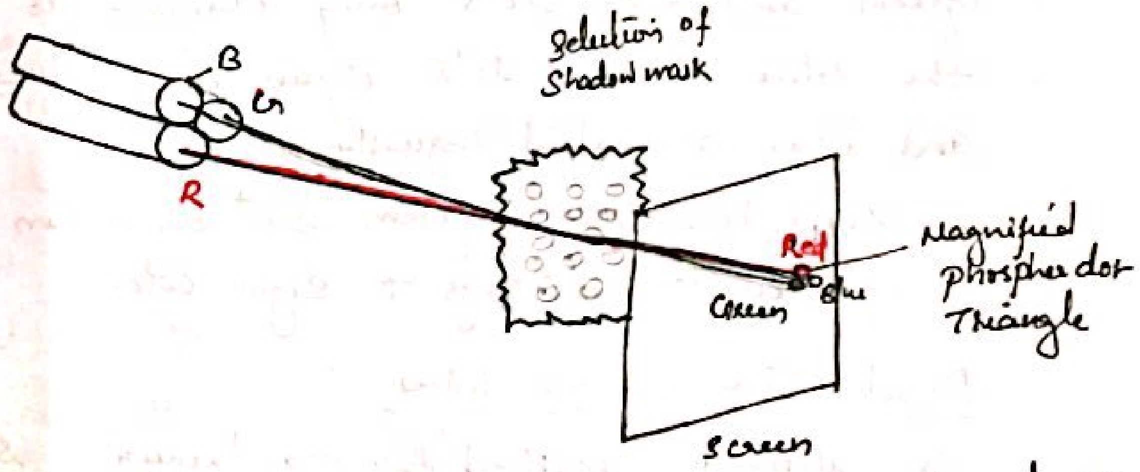
The beam penetration method for displaying color pictures has been used With random scan monitors. Two layers of phosphor usually red and green are coated on the inside of the CRT screen and the displayed color depends on the how far the electron beam penetrates into the phosphor layers.

A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, Combination of red and green light are emitted to show two additional colors, orange and yellow.

The speed of electrons and the screen color at any point is controlled by the beam acceleration voltage. Beam penetration has been an inexpensive way to produce color in random scan monitors. Only four colors are possible in this method and the quality of picture is not as good as with other methods.

Shadow Mask Method :-

This methods are commonly used in raster scan system (including color TV) because they produce a wider range of colors than the beam penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. one phospher dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT has three electron guns, one for each color dot, and a shadow mask grid just behind the phospher coated screen. The following figure shows the delta-delta shadow mask method. commonly used in color CRT system.



The three electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes. when the three beam pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask.

Another configuration for the three electron gun is an inline arrangement in which the three electron guns, and the corresponding red-green-blue color dot on the screen are aligned along one scan line instead of a triangular pattern. This type of configuration is used in high resolution color CRT.

Color variation in a shadow-mask CRT can be obtained by varying the intensity levels of the three electron beam. White or grey area is the result of activating all three dots with equal intensity. yellow is produced with the green and red dots only. Magenta is produced with the blue & red dots. cyan is produced when green and blue activated equally.

In some low-cost system electron beam is set to on or off, limiting displays to eight colors.

Direct View Storage Tubes

An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A direct view storage tube (DVST) stores the picture information as a charge distribution behind the phosphor screen. Two electron guns are used in the DVST. One, primary gun is used to store the picture pattern and the second the flood gun, maintains the picture display

Advantages:-

No refreshing is needed in DVST, very complex pictures can be displayed at very high resolutions without flicker.

Disadvantages:-

DVST do not display colors and selected parts of the picture cannot be erased. To eliminate the picture section, the entire screen must be erased and modified picture redrawn.

Flat Panel Displays

Flat panel display refer to a class of video devices that have reduced volume, weight and power requirement compared to CRT. Significant feature of flat panel display is that they are thinner than CRTs and we can hang them on walls or wear them on wrists.

Current uses for flat panel display include small TV monitors, calculators, pocket video games, laptop computers etc.

Flat panel displays are of two categories:-

i) Emissive displays

ii) Nonemissive displays.

The emissive displays are devices that convert electrical energy into light. eg:- plasma panels, thin-film electroluminescent display and light emitting diodes.

Nonemissive displays use optical effects to convert sunlight or light from other source into graphics pattern eg:- liquid crystal device.

plasma panels are also called gas discharge displays and are constructed by filling the region between two glass plates with a mixture of gases that includes neon. A series of vertical conducting ribbons is placed on one glass panel and a set of horizontal ribbons is built into the other glass panel. Voltage is applied to a pair of horizontal and vertical conductors cause the gas in the intersection of the two conductors to break down into a glowing plasma of electrons & ions. Picture definition is stored in a refresh buffer and the voltages are applied to refresh the pixel positions 60 times/sec. If the application of voltage is faster then the brightness of display is more. The Design of the plasma panel is shown in the following figure.

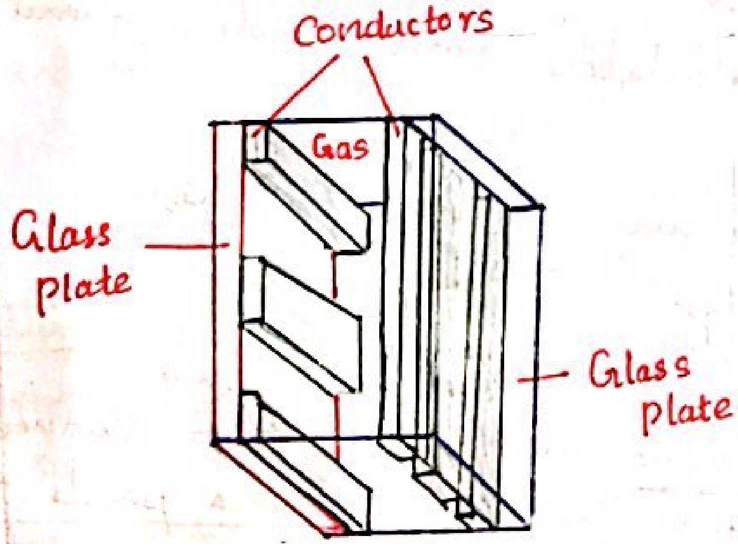


fig:- Basic design of a plasma-panel display device

Thin film electroluminescent display are similar in construction to a plasma panel. The only different is that the region between the glass plate is filled with a phospher. When sufficient voltage is applied to a pair of crossing electrodes, the phospher becomes a conductor in the area of the intersection of the two electrodes. Electroluminescent display require more power than plasma panel and good and grey color display are hard to acheive

Light Emitting Diode (LED)

In LED a matrix of diode is arranged to form the pixel position in the display the picture definition is stored in a refresh buffer. Information is read from the refresh buffer and converted to voltage levels are applied to the diodes to produce the light pattern in the display.

Liquid crystal Display (LCD)

LCD's are commonly used in small systems, such as calculator and laptop computers.

The non emissive devices LCD produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid crystal material that can be aligned to either block or transmit the light.

The term liquid crystal refers that the compounds have a crystalline arrangements of molecules, yet they flow like a liquid. Flat panel displays use nematic liquid crystal compounds that tend to keep the long axes of the Rod-Shaped molecules aligned.

A flat panel display can be constructed with a nematic liquid crystal as demonstrated in the following figure. Two glass plates, each containing a light polarizer at right angles to the other plate, Sandwich the liquid crystal material. Rows of horizontal transparent conductors are built into one glass plate, and the columns of vertical conductors are put into the other plate. The intersection of the two conductors defines the pixel position. Normally molecules are aligned in the 'on state'. polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is reflected back to the viewer

To turn off the pixel, a voltage is applied to the two intersecting conductors to align the molecules so that the light is not twisted. This type of flat-panel device is referred as a passive matrix LCD. picture definition are stored in a refresh buffer, and the screen is refreshed at the rate of 60frames/sec, as in the emissive devices.

Back lighting is also applied using solid state electronic devices, so that the system is not completely dependent on outside light sources. Colors can be displayed by using different materials or dyes and by placing a triad of color pixel at each screen location.

Another method of constructing LCD is to place a transistor at each pixel location using thin-film transistor technology.

The transistors are used to control the voltages at pixel locations and to prevent charge from gradually leaking out of the liquid crystal cells. These devices are called active matrix displays

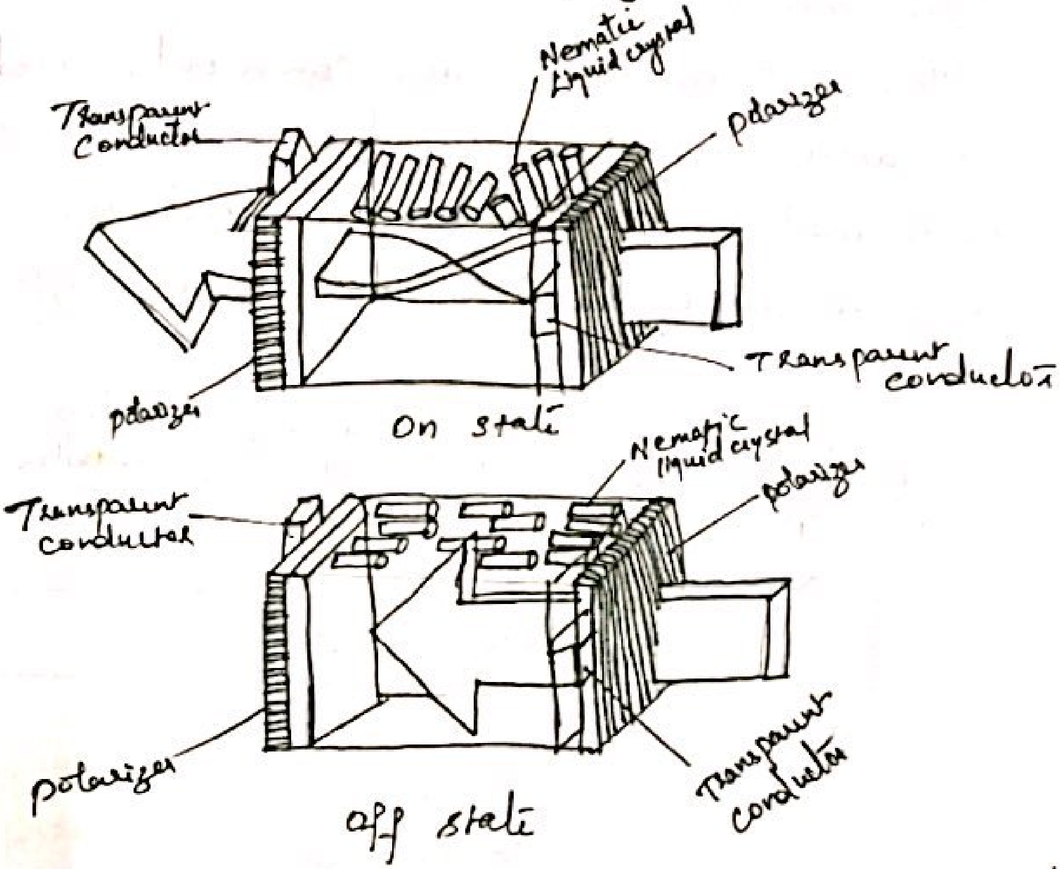
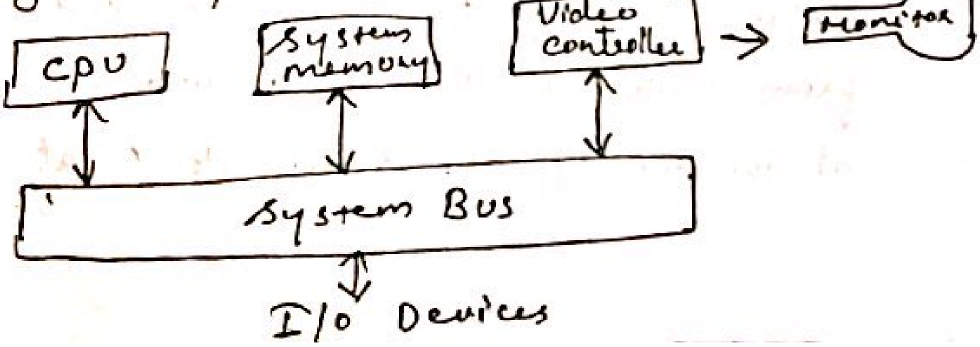


fig: Light twisting, Shutter effect used in the design of most liquid crystal display devices

Raster Scan systems

Interactive raster graphics systems consist of a special purpose processor called the video controller and display controller used to control the operation of the display device. organization of the simple raster systems is shown below



The frame buffer can be anywhere in the system memory and the video controller accesses the frame buffer to refresh the screen.

Video Controller

Following figure shows the commonly used organization for raster system. A fixed area of the system memory is reserved for the frame buffer and the video controller is given direct access to the frame buffer memory.

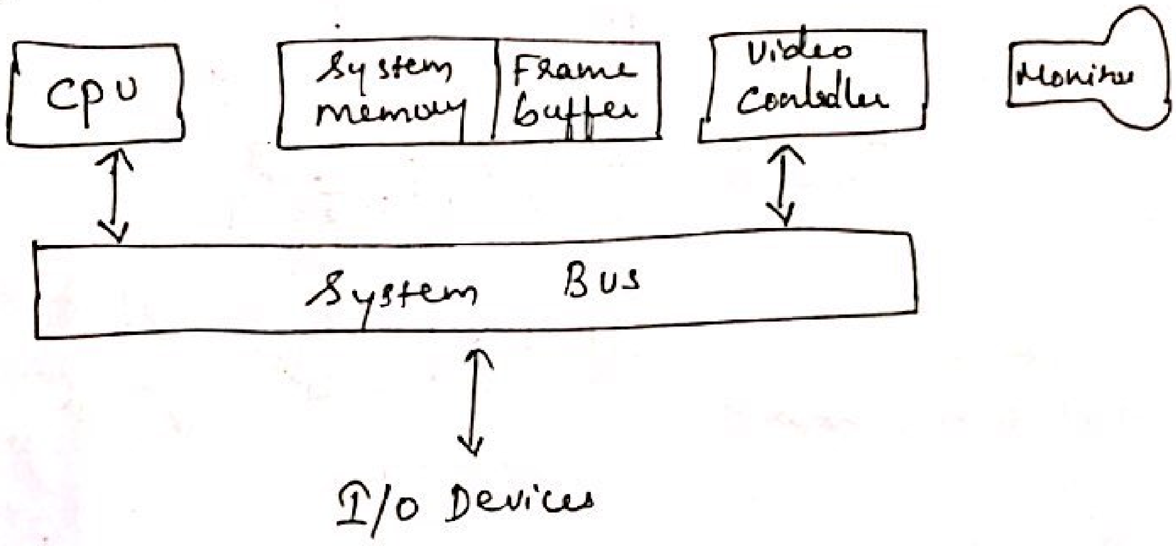
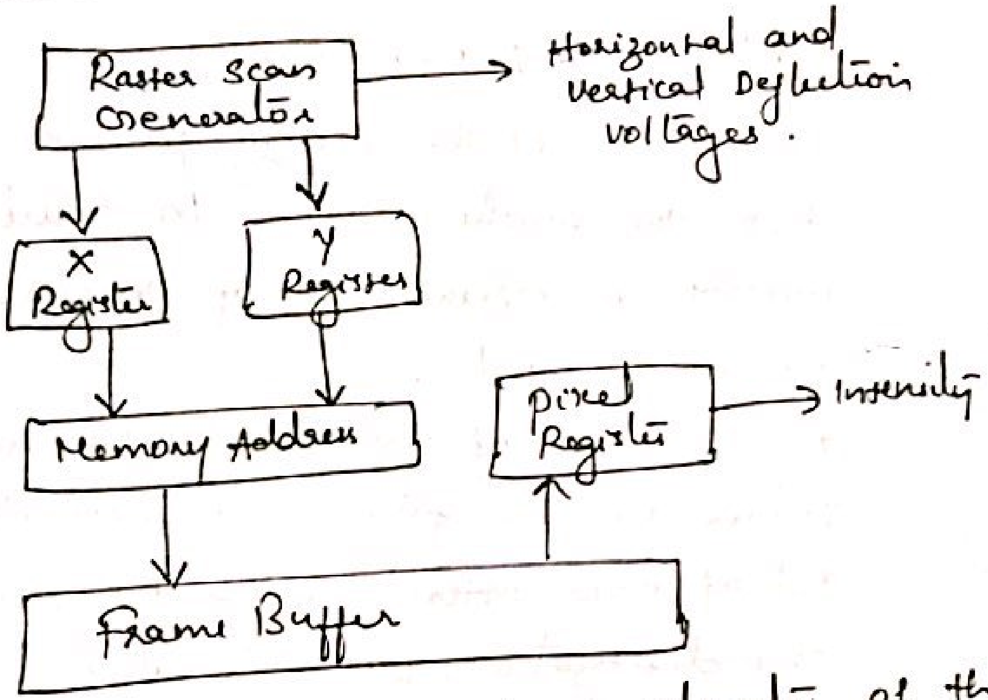


fig: Architecture of raster systems.

Frame buffer locations, and the corresponding screen positions are referenced in Cartesian coordinates. For many graphics monitors, the coordinate origin is defined at the lower left screen corner. The screen surface is represented as the first quadrant of a 2-D system with positive x value increasing to the right and positive y value increasing from bottom to top. Scanlines are labeled from ymax at the top of the screen to 0 at the bottom. Along each scan line screen pixel positions are labeled from 0 to x_{max}.

The basic refresh operations of the video controller are given below.



Two Registers are used to store the coordinates of the screen pixels. Initially the x register is set to 0 and y register is set to y_{max}. The value stored in the frame buffer for this pixel position is then retrieved and used to set the intensity of the CRT beam. The x register is incremented by 1, and the process repeated for the next pixel on the top scan line. This procedure is repeated for each pixel along the scan line. After the last pixel on the top scan line has been processed, the x register is reset to 0 and the y register is decremented by 1. pixels along this scan line are then processed in turn, and the procedure is repeated for each successive scan line

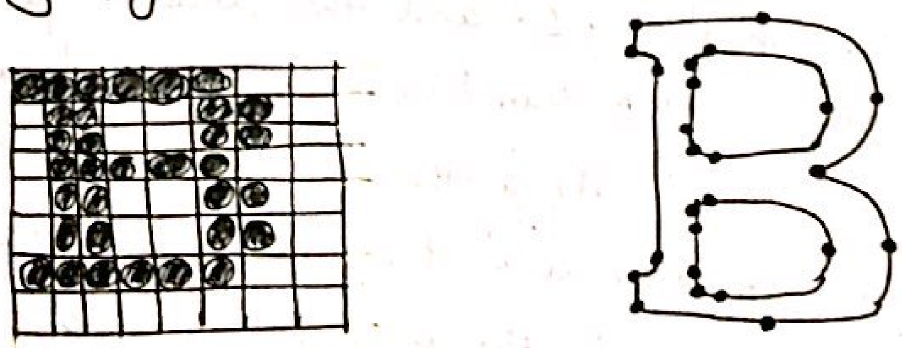
After cycling through all pixels along the bottom scan line (Y=0) the video controller resets the register to the first pixel position on the top scan line and the refresh process starts over.

Raster-Scan Display Processor

The purpose of the display processor is to free the CPU from the graphics chores. In addition to the system memory, a seperate display processor memory area can also be provided.

A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is called scan conversion. Graphics Commands specifying straight lines and other geometric objects are scan converted into a set of discrete intensity points.

Characters can be defined with rectangular grids or they can be defined with curved outlines as shown in the following figure

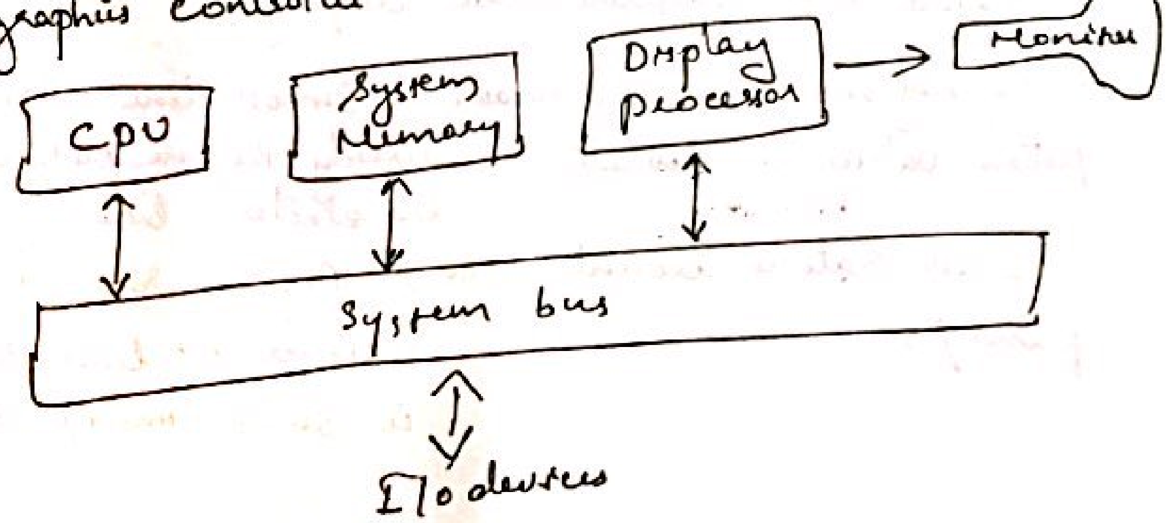


The array size for character grids can vary from about 5 by 7 to 9 by 12 or more for higher quality displays

Display processor is also designed to perform a number of additional operation. These functions include generating various line styles (dashed, dotted or solid) displaying color areas, and performing certain transformations and manipulations on displayed objects. Display processors are designed to interface with interactive input devices, such as a mouse

Randomscan Systems

The organization of a simple random-scan (vector) system is shown in the following diagram. An application program is input and stored in the system memory along with a graphics package. Graphics commands in the application program are translated by the graphics package into a display file stored in the system m/m. This display file is then accessed by the display processor to refresh the screen. The display processor cycles through each command in the display file program once during every refresh cycle. The display processor in a random scan system is referred as a display processing unit or a graphics controller



Graphics patterns are drawn on a random scan system by directing the electron beam along the component lines of the picture. Lines are defined by the values for their coordinate endpoints and these input coordinate values are converted to x and y deflection voltage. A scene is then drawn one line at a time by positioning the beam to fill in the line between specified endpoints.

Difference between Raster Scan and Random scan System

Raster Scan	Random scan
1) Raster scan display has low resolution as picture definition is stored as an intensity value	1) Random scan display has high resolution as it stores picture definition as a set of command lines
2) Electron beam is directed from top to bottom and one row at a time on whole screen	2) The electron beam is directed to only that part of screen where picture is required to be drawn, one line at a time.
3) Real life images can be displayed with different shades	3) Real life images cannot be displayed
4) Zig-zag line is produced because plotted values are discrete	4) Smooth line is produced because directly the line path is followed by electron beam.
5) Refresh rate is around 60-80 frames / sec	5) Refresh rate depends on the number of lines to be displayed i.e. 30-60 times/sec.

Points and lines

Point is the fundamental element of picture representation. Point is the position in the plan defined as either pair or triplets of number depending upon the dimension.

Two points represents a line or Edge. Three or more points represent a polygon.

Curved lines are represented by the short straight lines.

In a Random-scan (vector) system point-plotting instruction are stores in the display list, and coordinate values in these instructions are converted to deflection voltages that position the electron beam at the screen locations to be plotted during each refresh cycle.

In Black and white Raster system, a point is plotted by setting the bit value corresponding to a specified screen position within the frame buffer to 1. As the electron beam sweeps across each horizontal scan line, it emits a burst of electrons (plots a point) whenever a value of 1 is encountered in the frame buffer.

In an RGB System, the frame buffer is loaded with the color codes for the intensities that are to be displayed at the screen pixel positions.

Line drawing is accomplished by calculating intermediate positions along the line path between two specified endpoint positions. An output device is directed to fill in these positions between endpoints.

For analog devices, a straight line can be drawn smoothly from one endpoint to the other. Linearly varying horizontal and vertical deflection voltages are generated that are proportional to the required changes in the x and y directions to produce smooth line.

Digital devices display straight line segments by plotting discrete points between two endpoints. Discrete coordinate position along the line path are calculated from the equation of the line.

To load an intensity value into the frame buffer at a position (x,y) the procedure form is setpixel (x, y, intensity)

To retrieve the current frame buffer intensity for a specified location the function is getpixel (x, y).

Line Drawing Algorithms

The equation for straight line is

$$y = mx + b$$

where 'm' is the slope of the line
'b' is the y intercept.

The two endpoints of a line segments are at the positions (x_1, y_1) and (x_2, y_2). The value for the slope m and y-intercept b can be calculated as

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

$$b = y_1 - m \cdot x_1$$

For any given x interval Δx along a line, the corresponding y interval Δy can be computed by

$$\Delta y = m \Delta x \longrightarrow ①$$

Similarily x interval Δx corresponding to a specified Δy as

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

Now if $\Delta x = 1$ ie $x_{i+1} - x_i = 1$ then ① becomes

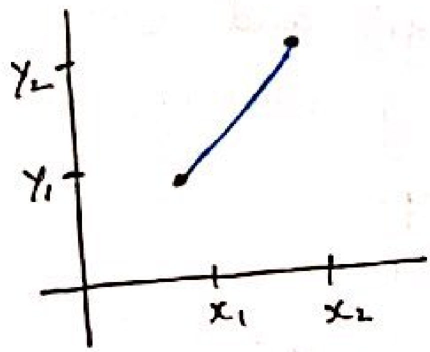
$$\Delta y = m. \quad \text{ie} \quad y_{i+1} - y_i = m \quad \text{ie} \quad y_{i+1} = y_i + m.$$

Thus a unit change in 'x' changes 'y' by 'm' which is a constant for a given line.

DDA Algorithm

* The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy and Δx.
* The line is sample at unit intervals in one coordinate and determine corresponding integer values nearest the line path for other coordinate.

Consider a line with positive slope as shown in the following figure



If $m \leq 1$ then line is sample at unit x interval and the successive y-value will be calculated as

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

Subscript k takes integer value starting from 1 and increases by 1 until reaches end point.

'm' is a slope which is a real number between 0 to 1.

If $m \geq 1$ then line is sample at unit y interval and the succeeding x value is calculated as

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

Derivation of Equation in DDA Algorithm

DDA Algm is used to find the intermediate points between the end points of the line by calculating Δx and Δy with the use of slop 'm'.

In the above given line in the screen each pixel or point can be calculated interms of a coordinates

The slope of the line is given by

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

Let the start point be (x_k, y_k) and the end point be (x_{k+1}, y_{k+1})

Then the slope $m = \dfrac{y_{k+1} - y_k}{x_{k+1} - x_k}$

Scanned with CamScanner

Case - 1 [m<1]

x is change in Unit interval

ie $\boxed{x_{k+1} = x_k + 1}$

ie $x_{k+1} - x_k = 1$

$\therefore m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$ becomes $\frac{y_{k+1} - y_k}{1}$

ie $m = y_{k+1} - y_k$

$\boxed{\therefore y_{k+1} = y_k + m}$

Case - 2 [m>1]

y change in Unit interval

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

ie $y_{k-1} - y_k = 1$

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

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

Case - 3 (m=1)

x and y change in Unit interval

ie $\boxed{\begin{array}{l} x_{k+1} = x_k + 1 \\ y_{k+1} = y_k + 1 \end{array}}$

Algorithm :-

1. Calculate slope m
2. If m<1 then x changes in Unit interval and

y moves in deviation

$(x_{k+1}, y_{k+1}) = (x_k+1, y_k+m)$

3. If $m > 1$ then x moves with deviation and y moves in unit intervals

$(x_{k+1}, y_{k+1}) = (x_k + \frac{1}{m}, y_k+m)$

4. If $m = 1$ then x and y moves in unit intervals

$(x_{k+1}, y_{k+1}) = (x_k+1, y_k+1)$

Q plot the line using DDA algorithm having initial point as (5,4) and endpoint as (12,7)

$(x_1, y_1) = (5,4)$

$(x_k, y_k) = (12,7)$

$\Delta x = x_k - x_1 = 12-5 = 7$

$\Delta y = y_k - y_1 = 7-4 = 3$

$m = \frac{\Delta y}{\Delta x} = \frac{3}{7} = 0.4 < 1$

Step $= 7$ (abs(Δx)) since abs(Δx) > abs (Δy)

$X_{increment} = \frac{\Delta x}{step} = \frac{7}{7} = 1$

$Y_{increment} = \frac{\Delta y}{step} = \frac{3}{7} = 0.4$

The points of x and y are given below:

X	Y	Round(Y)
5	4	4
6	4.4	4
7	4.8	5
8	5.2	5

X	Y	Round(Y)
9	5.6	6
10	6	6
11	6.4	6
12	6.8	7

Q plot the line using DDA algorithm having initial point as (5,7) and end point as (10,15)

$(x_1, y_1) = (5,7)$
$(x_k, y_k) = (10,15)$

$\Delta x = x_k - x_1 = 10-5 = 5$

$\Delta y = y_k - y_1 = 15-7 = 8$ step = 8

$m = \frac{\Delta y}{\Delta x} = \frac{8}{5} > 1$

$x_{increment} = \frac{\Delta x}{Step} = \frac{5}{8} = 0.6$

$y_{increment} = \frac{\Delta y}{Step} = \frac{8}{8} = 1$

X	Round(x)	Y
5	5	7
5.6	6	8
6.2	6	9
6.8	7	10
7.4	7	11
8	8	12
8.6	9	13
9.2	9	14
10	9.8	15

Q) plot the line using DDA algorithm having initial point as (12,9) and end point as (17,14)

$(x_1, y_1) = (12, 9)$

$(x_k, y_k) = (17, 14)$

$\Delta x = x_k - x_1 = 17 - 12 = 5$

$\Delta y = y_k - y_1 = 14 - 9 = 5$

$m = \frac{\Delta y}{\Delta x} = \frac{5}{5} = 1$ step = 5

$x_{increment} = \frac{5}{5} = 1$

$y_{increment} = \frac{5}{5} = 1$

x	Y
12	9
13	10
14	11
15	12
16	13
17	14

Suppose $(x_1, y_1) = (17, 14)$

$(x_k, y_k) = (12, 9)$

$\Delta x = 12 - 17 = -5$ step = 5

$\Delta y = 9 - 14 = -5$

$m = \frac{-5}{-5} = 1$

$x_{increment} = \frac{5}{-5} = -1$

$y_{increment} = \frac{5}{-5} = -1$

x	y
17	14
16	13
15	12
14	11
13	10
12	9

Advantages of DDA Algorithm

① DDA Algorithm is a faster method for calculating pixel position than the direct use of line equation

② DDA algorithm eliminates multiplication

Disadvantage of DDA Algorithm

① DDA Algorithm does not produce a smooth line

② Rounding function in DDA algorithms need extra Computation

Scanned with CamScanner

```
procedure lineDDA (xa, ya, xb, yb : integer);
var
  dx, dy, steps, k : integer
  xincrement, yincrement, x, y : real;
begin
  dx := xb - xa;
  dy := yb - ya;
  if abs(dx) > abs(dy) then step := abs(dx)
  else steps := abs(dy);
  xincrement := dx / steps;
  yincrement := dy / steps;
  x := xa;
  y := ya;
  setpixel (round(x), round(y), 1);
  for k = 1 to steps do
     begin
       x := x + xincrement;
       y := y + yincrement;
       setpixel (round(x), round(y), 1);
     end
  end; (line DDA)
```

Bresenham's Line Algorithm

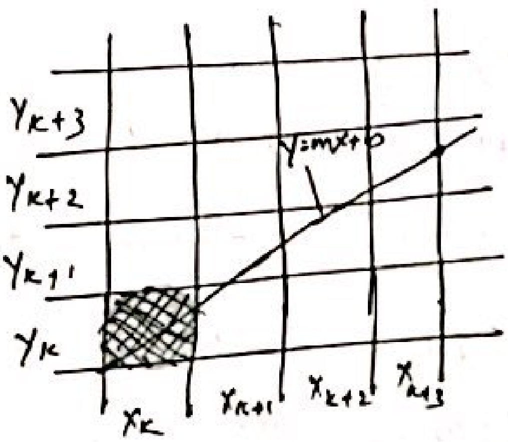
Bresenham's line algorithm is an accurate and efficient raster line generating Algorithm. using only incremental integer calculations

Consider the scan conversion process for lines with positive slope less than 1 $(m<1)$. pixel positions along a line path are then determined by sampling at unit x intervals. Starting from the left endpoint (x_0, y_0) of a given line

Scanned with CamScanner

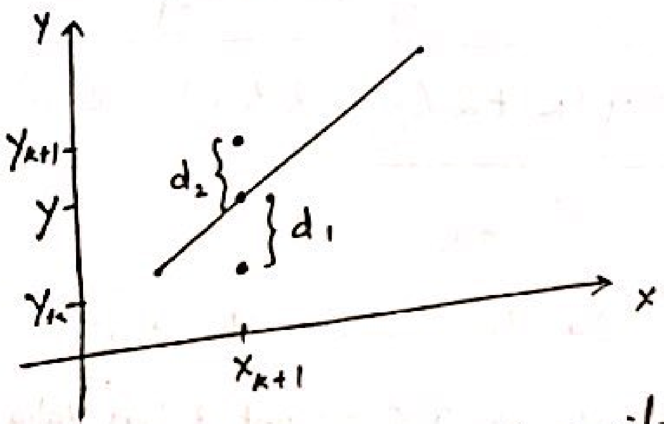
plot the pixel for each successive x position and whose scan-line y is closest to the line path.

Consider the following figure.



Assume that initial pixel is plotted in (x_k, y_k) coordinate and we need to decide the next pixel to be plotted is either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1})

To determine y-value sampling is performed in the vertical line path. as d_1 and d_2



The y coordinate at the pixel position $x_k + 1$ is calculated by

$$y = m(x_k + 1) + b \longrightarrow ①$$

Scanned with CamScanner

Then

$$d_1 = y - y_k$$

Sub ① in d_1

$$d_1 = m(x_k+1) + b - y_k$$

and $d_2 = (y_k+1) - y$

$$= y_k + 1 - m(x_k+1) - b$$

$$d_1 - d_2 = m(x_k+1) + b - y_k - y_k - 1 + m(x_k+1) + b$$

$$d_1 - d_2 = 2m(x_k+1) - 2y_k + 2b - 1 \longrightarrow ②$$

Multiply Δx on both of equation ②

$$\Delta x(d_1 - d_2) = \Delta x[2m(x_k+1) - 2y_k + 2b - 1] \longrightarrow ③$$

Sub $m = \frac{\Delta y}{\Delta x}$ in equ ③

$$\Delta x(d_1 - d_2) = \Delta x\left[2\frac{\Delta y}{\Delta x}(x_k+1) - 2y_k + 2b - 1\right]$$

$$= 2\Delta y\, x_k + 2\Delta y - 2y_k \Delta x + \Delta x(2b-1)$$

$$\boxed{P_k = 2\Delta y x_k + 2\Delta y - 2y_k \Delta x + \Delta x(2b-1)}$$

or

$P_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$

where c is a constant & has value $2\Delta y + \Delta x(2b-1)$

To find P_{k+1} sub $x_k = x_{k+1}$ and $y_k = y_{k+1}$ in equ P_k

$$P_{k+1} = 2\Delta y\, x_{k+1} + 2\Delta y - 2y_{k+1}\Delta x + \Delta x(2b-1)$$

$$p_{k+1} - p_k = 2\Delta y x_{k+1} - 2\Delta x\, y_{k+1} + 2\Delta y + \Delta x(2b-1)$$
$$-\left(2\Delta y\, x_k - 2y_k \Delta x + 2\Delta y + \Delta x(2b-1)\right)$$
$$= 2\Delta y x_{k+1} - 2\Delta x\, y_{k+1} + \cancel{2\Delta y} + \cancel{\Delta x(2b-1)}$$
$$- 2\Delta y x_k + 2y_k\Delta x - \cancel{2\Delta y} - \cancel{\Delta x(2b-1)}$$
$$= 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

Sub $x_{k+1} = x_k + 1$ in the above equatn

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

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

Initial decision parameter p_0 can be calculated from p_k with initial point (x_k, y_k)

$$p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2b-1)$$

sub $b = y - mx$

$$p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2(y - mx) - 1)$$
$$- 2\Delta x m x - \Delta x$$
$$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y$$

Sub $y = y_k$, $x = x_k$ and $m = \frac{\Delta y}{\Delta x}$

$$p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta x\frac{\Delta y}{\Delta x}x_k - \Delta x$$
$$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x$$
$$= 2\Delta y x_k + 2\Delta y -$$

$\therefore$ Initial decision parameter $\boxed{p_0 = 2\Delta y - \Delta x}$

Scanned with CamScanner

If $p_k < 0$ then the next coordinate is x_{k+1}, y_k

so the p_{k+1} equation becomes

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

$$\Downarrow$$

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

$y_{k+1} - y_k = 0.$
so $2\Delta x(y_{k+1} - y_k) = 0$

If $p_k > 0$ then the next coordinate is x_{k+1}, y_{k+1}

$$\boxed{p_{k+1} = p_k + 2\Delta y - 2\Delta x}$$

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

<u>Bresenhams' Line Drawing Algorithm for $|m| < 1$</u>

1. Input the two line endpoints and store the left endpoint in (x_0, y_0)
2. Load (x_0, y_0) into the frame buffer, that is, plot the first point.
3. Calculate constants $\Delta x, \Delta y, 2\Delta y$ and $2\Delta y - 2\Delta x$ an obtain the starting value for the decision parameter as $p_0 = 2\Delta y - \Delta x$
4. At each x_k along the line, starting at $k = 0$, perform the following test:
   If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and.
   $$p_{k+1} = p_k + 2\Delta y$$
   Otherwise the next point to plot is $(x_k + 1, y_k + 1)$ and
   $$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$
5. Repeat step 4 Δx times

```
procedure lineBres (xa, ya, xb, yb : integer);
var
  dx, dy, x, y, xEnd, p : integer;
begin
  dx := abs(xa - xb);
  dy := abs(ya - yb);
  p := 2 * dy - dx;
  { determine which point to use as start, which as
    end }
  if xa > xb then
    begin
      x = xb ;
      y = yb ;
      xEnd := xa
  end {if xa > xb}
  else
    begin
      x := xa ;
      y := ya ;
      xEnd = xb
    end;
  Setpixel (x, y, 1);
  while x < xEnd do
    begin
      x := x + 1 ;
      if p < 0 then p := p + 2 * dy
      else
        begin
          y := y + 1
          p := p + 2 * (dy - dx)
        end ;
      Setpixel (x, y, 1)
    end;
end ; { lineBres }
```

Scanned with CamScanner

Q Digitize the line with endpoints (20,10) and (30,18) using Bresenham's line drawing Algorithm. The line has a slope of 0.8 with $\Delta x = 10$ $\Delta y = 8$

$X_a = 20$

$X_b = 30$

$X_a < X_b$

So $X_{start} = 20$

$X_{End} = 30$

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

$= 2 \times 8 - 10 = 16 - 10 = \underline{\underline{6}}$

$2\Delta y = \underline{\underline{16}}$ and $2\Delta y - 2\Delta x = 16 - 20 = \underline{\underline{-4}}$

Initial point to be plotted $(x_0, y_0 = 20, 10)$

K	P_K	(x_{k+1}, y_{k+1})	
0	6	(21, 11)	$P_K > 0$ So $P_{k+1} = P_k + 2\Delta y - 2\Delta x = 6 - 4 = 2$
1	2	(22, 12)	$P_k > 0$ $P_{k+1} = 2 - 4 = -2$
2	-2	(23, 12)	$P_k < 0$ $P_{k+1} = P_k + 2\Delta y = -2 + 16 = 14$
3	14	(24, 13)	$P_k > 0$ $P_{k+1} = 14 - 4 = 10$
4	10	(25, 14)	$P_k > 0$ $P_{k+1} = 10 - 4 = 6$
5	6	(26, 15)	$P_K > 0$ $P_{k+1} = 6 - 4 = 2$
6	2	(27, 16)	$P_k > 0$ $P_{k+1} = \cancel{6}\,2 - 4 = -2$
7	-2	(28, 16)	$P_k < 0$ $P_{k+1} = -2 + 16 = 14$
8	14	(29, 17)	$P_k > 0$ $P_{k+1} = 14 - 4 = 10$
9	10	(30, 18)	$P_k > 0$ $P_{k+1} = 10 - 4 = 6$

The End given is (30,18). So the different coordinates in the straight line is the above given (x_{k+1}, y_{k+1}) Coordinates

Q Draw a line between (5,12) and (15,20) using Bresenham's algorithm

$\Delta x = 10 \quad (15-5) = 10$

$\Delta y = 20 - 12 = 8$

$2\Delta y = 16$

$2\Delta y - 2\Delta x = 16 - 20 = -4$

$p_0 = 2\Delta y - \Delta x$
$= 16 - 10 = 6$

$(x_0, y_0) = 5, 12$

K	p_K	(x_{K+1}, y_{K+1})	
0	6	(6, 13)	$p_K > 0 \quad p_{K+1} = p_K + 2\Delta y - 2\Delta x = 6 - 4 = 2$
1	2	(7, 14)	$p_K > 0 \quad p_{K+1} = 2 - 4 = -2$
2	-2	(8, 14)	$p_K < 0 \quad p_{K+1} = p_K + 2\Delta y = -2 + 16 = 14$
3	14	(9, 15)	$p_K > 0 \quad p_{K+1} = 14 - 4 = 10$
4	10	(10, 16)	$p_K > 0 \quad p_{K+1} = 10 - 4 = 6$
5	6	(11, 17)	$p_K > 0 \quad p_{K+1} = 6 - 4 = 2$
6	2	(12, 18)	$p_K > 0 \quad p_{K+1} = 2 - 4 = -2$
7	-2	(13, 18)	$p_K < 0 \quad p_{K+1} = -2 + 16 = 14$
8	14	(14, 19)	$p_K > 0 \quad p_{K+1} = 14 - 4 = 10$
9	10	(15, 20)	

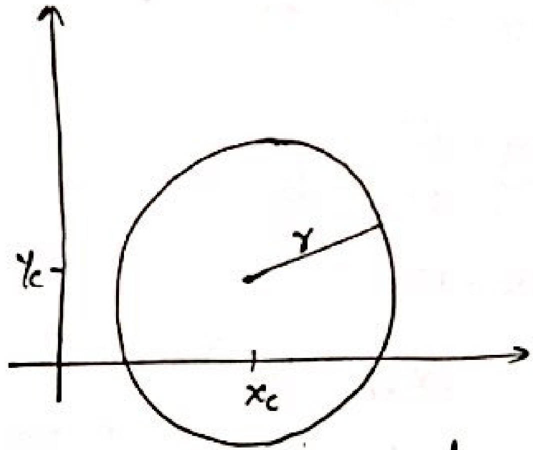
The end point given is (15, 20)

Circle generation Algorithm

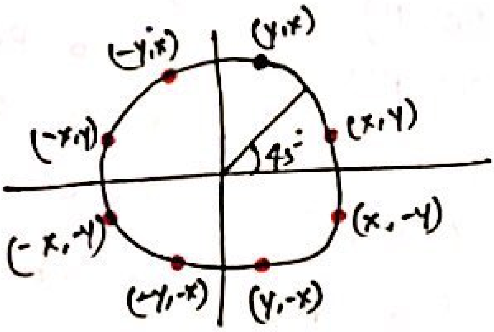
A circle is defined as the set of points that are all at a given distance r from a center position (x_c, y_c). The distance relationship is expressed by the pythagorean theorem in Cartesian coordinates as

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

The above equation is used to calculate the positions of points on a circle by stepping x axis in unit intervals



The symmetry property of a circle can be used to generate the points in each of the quadrant. The symmetry property of the circle is given below.



All pixel positions around a circle can be calculating only the points within the sector from $x = 0$ to $x = y$

Midpoint circle Algorithms

As in the raster line algorithm, sampling is done at unit intervals and determine the closest pixel position to the specified circle path at each step.

For a given radius 'r' and screen center position (x_c, y_c), first calculate the pixel position around a circle path centered at the coordinate origin (0,0). Then each calculated position (x, y) is moved to its proper screen position by adding x_c to x and y_c to y.

Along the circle section from $x=0$ to $x=y$ in the first quadrant the slope of the curve varies from 0 to -1. So we can take unit steps in the positive x direction over this octant and use a decision parameter to determine which of the two possible y' position is closer to the circle path at each step.

The circle function can be defined by

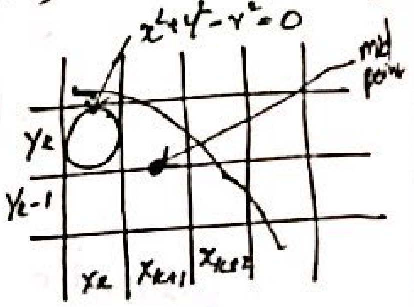
$$f_{circle}(x,y) = x^2 + y^2 - r^2$$

Any point (x,y) on the boundary of the circle with radius 'r' satisfies the equation $f_{circle}(x,y) = 0$. If the point is in the interior of the circle, the circle function is negative. And if the point is outside the circle, the circle function is positive. The relative position of any point (x,y) can be determined by checking the sign of the circle function

$$f_{circle}(x,y) \begin{cases} <0, & \text{if } (x,y) \text{ is inside the circle boundary} \\ =0, & \text{if } (x,y) \text{ is on the circle boundary} \\ >0, & \text{if } (x,y) \text{ is outside the circle boundary} \end{cases}$$

The following figure shows the midpoint between the two candidate pixels at sampling position x_k+1. Assume that pixel is plotted at the position (x_k, y_k) and need to determine the next pixel position is either (x_k+1, y_k) or (x_k+1, y_k-1) by using decision parameter.

Decision parameter can be calculated using the midpoint of the pixels (x_k+1, y_k) and (x_k+1, y_k+1) in the circle function. $f_{circle}(x,y) = x^2+y^2-r^2$



$$P_k = f_{circle}\left(x_k+1,\ y_k-\frac{1}{2}\right)$$

$$\boxed{P_k = (x_k+1)^2 + \left(y_k-\frac{1}{2}\right)^2 - r^2}$$

$$P_{k+1} = \left(x_{k+1}+1\right)^2 + \left(y_{k+1}-\frac{1}{2}\right)^2 - r^2$$

$$P_{k+1}-P_k = \left(x_{k+1}+1\right)^2 + \left(y_{k+1}-\frac{1}{2}\right)^2 - r^2 - (x_k+1)^2 - \left(y_k-\frac{1}{2}\right)^2 + r^2$$

sub $x_{k+1} = x_k+1$

$$= \left((x_k+1)+1\right)^2 + \left(y_{k+1}-\frac{1}{2}\right)^2 - (x_k+1)^2 - \left(y_k-\frac{1}{2}\right)^2$$

$$= (x_k+1)^2 + 2(x_k+1) + 1 + y_{k+1}^2 - y_{k+1} + \frac{1}{4} - (x_k+1)^2 - y_k^2 + y_k - \frac{1}{4}$$

$$= 2(x_k+1) + \left(y_{k+1}^2 - y_k^2\right) - \left(y_{k+1}-y_k\right) + 1$$

$$P_{k+1} - P_k = 2(x_k+1) + \left(y_{k+1}^2 - y_k^2\right) - \left(y_{k+1}-y_k\right) + 1$$

$$\boxed{\therefore P_{k+1} = P_k + 2(x_k+1) + \left(y_{k+1}^2 - y_k^2\right) - \left(y_{k+1}-y_k\right) + 1}$$

The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$

$P_k = (x_k+1)^2 + (y_k - \frac{1}{2})^2 - r^2$

sub $(0, r)$ for (x_k, y_k) in equation P_k

$\therefore P_0 = (0+1)^2 + (r - \frac{1}{2})^2 - r^2$

$= 1 + r^2 - r + \frac{1}{4} - r^2$

$\boxed{\therefore P_0 = \frac{5}{4} - r}$ or $\boxed{P_0 = 1 - r}$

If $P_k < 0$ then $y_{k+1} = y_k$ $\therefore$ next coordinate becomes (x_{k+1}, y_k). The P_{k+1} equation then becomes

$P_{k+1} = P_k + 2(x_k+1) + (y_k^2 - y_k^2) - (y_k - y_k) + 1$

$\boxed{\therefore P_{k+1} = P_k + 2(x_k+1) + 1 \quad \text{if } P_k < 0}$

If $P_k > 0$ then $y_{k+1} = y_k - 1$ next coordinate becomes (x_{k+1}, y_{k-1}). The P_{k+1} equation then becomes

$P_{k+1} = P_k + 2(x_k+1) + ((y_k-1)^2 - y_k^2) - (y_k - 1 - y_k) + 1$

$= P_k + 2(x_k+1) + y_k^2 - 2y_k + 1 - y_k^2 - y_k + 1 + y_k + 1$

$= P_k + 2(x_k+1) - 2y_k + 1.$

$= \cancel{P_k + 2(x_k+1)}$

$P_{k+1} = P_k + 2(x_k+1) + 1 - 2y_k + 2$

$= P_k + 2(x_k+1) - 2(y_k+1) + 1$

$\boxed{P_{k+1} = P_k + 2((x_k+1) - (y_k+1)) + 1}$

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$
2. Calculate the initial value of the decision parameter as $p_0 = \frac{5}{4} - r$
3. At each x_k position, starting at $k=0$, perform the following test: If $p_k < 0$, the next point along the circle centered on $(0,0)$ is (x_k+1, y_k) and

   $$p_{k+1} = p_k + 2x_{k+1} + 1$$

   Otherwise the next point along the circle is (x_k+1, y_k-1)

   and $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$

   where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.
4. Determine symmetry points in the other seven octants
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

   $$x = x + x_c \quad ; \quad y = y + y_c$$
6. Repeat step 3 through 5 until $x \geq y$.

```
Procedure circleMidpoint (xcenter, ycenter, radius : integer);
var
  p, x, y: integer;
  procedure plot points;
begin
  setpixel (xcenter + x , ycenter + y, 1);
  Set pixel ( x center - x , ycenter + y , 1);
  Set pixel ( x center + x , ycenter - y, 1);
```

```
    Setpixel (xcenter - x, ycenter - y, 1);
    Setpixel (xcenter + y, ycenter + x, 1);
    Setpixel (xcenter - y, ycenter + x, 1);
    Setpixel (xcenter + y, ycenter - x, 1);
    Setpixel (xcenter - y, ycenter - x, 1)
end; plot points
begin
 x := 0;
 y := radius;
 plotpoints;
 p = 1 - radius;
 while x < y do
    begin
     if p < 0 then
        x := x + 1
     else
        begin
         x := x + 1;
         y := y - 1
        end;
    if p < 0 then
       p := p + 2 * x + 1
    else
       p := p + 2 * (x - y) + 1
    plot points
  end;
end; {circle midpoint}
```

Q. Given a circle with radius $r = 10$, determine the first quadrant positions along the circle octant in the first quadrant from $x = 0$ to $x = y$ by using midpoint circle algorithm.

The initial value of the decision parameter is

$p_0 = 1 - r = -9$

Initial point is $(x_0, y_0) = (0, 10)$

Successive decision parameter values and positions along the circle path are calculated as

K	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	p_{k+1}
0	-9	(1, 10)	2	20	$p_{k+1} = p_k + 2x_{k+1} + 1$ $= -9 + 2 + 1 = -6$
1	-6	(2, 10)	4	20	$p_{k+1} = -6 + 4 + 1 = -1$
2	-1	(3, 10)	6	20	$p_{k+1} = -1 + 6 + 1 = 6$
3	6	(4, 9)	8	18	$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$ $= 6 + 8 + 1 - 18 = -3$
4	-3	(5, 9)	10	18	$p_{k+1} = -3 + 10 + 1 = 8$
5	8	(6, 8)	12	16	$p_{k+1} = 8 + 12 + 1 - 16 = 5$
6	5	(7, 7)	14	14	

Since $x \geq y$ ie $7 = 7$ so we can stop finding coordinates. The different points on the Quadrants are given below

$\therefore Q_1 (x, y)$	$Q_2 (-x, y)$	$Q_3 (-x, -y)$	$Q_4 (x, -y)$
(0, 8)	(0, 8)	(0, -8)	(0, -8)
(1, 10)	(-1, 10)	(-1, -10)	(1, -10)
(2, 10)	(-2, 10)	(-2, -10)	(2, -10)
(3, 10)	(-3, 10)	(-3, -10)	(3, -10)
(4, 9)	(-4, 9)	(-4, -9)	(4, -9)
(5, 9)	(-5, 9)	(-5, -9)	(5, -9)
(6, 8)	(-6, 8)	(-6, -8)	(6, -8)
(7, 7)	(-7, 7)	(-7, -7)	(7, -7)
(8, 6)	(-8, 6)	(-8, -6)	(8, -6)
(9, 5)	(-9, 5)	(-9, -5)	(9, -5)
(9, 4)	(-9, 4)	(-9, -4)	(9, -4)
(10, 3)	(-10, 3)	(-10, -3)	(10, -3)
(10, 2)	(-10, 2)	(-10, -2)	(10, -2)
(10, 1)	(-10, 1)	(-10, -1)	(10, -1)
(8, 0)	(-8, 0)	(-8, 0)	(8, 0)

Q Determine the position of the circle in the first quadrant With radius = 8 and the initial points are (0,8) ?

$P_0 = 1 - r = 1 - 8 = -7$

Initial point is $(x_0, y_0) = (0,8)$

K	P_K	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	P_{k+1}
0	-7	(1, 8)	2	16	$P_{k+1} = -7+2+1 = -4$
1	-4	(2, 8)	4	16	$P_{k+1} = -4+4+1 = 1$
2	1	(3, 7)	6	14	$P_{k+1} = 1+6+1-14 = -6$
3	-6	(4, 7)	8	14	$P_{k+1} = -6+8+1 \cancel{14} = 3$
4	3	(5, 6)	10	12	$P_{k+1} = 3+10+1-12 = 2$
5	2	(6, 5)	12	10	$P_{k+1} = 2+12+1-10 = 5$

Q_1 (x, y)	Q_2 (-x, y)	Q_3 (-x, -y)	Q_4 (x, -y)
(0, 8)	(0, 8)	(0, -8)	(0, -8)
(1, 8)	(-1, 8)	(-1, -8)	(1, -8)
(2, 8)	(-2, 8)	(-2, -8)	(2, -8)
(3, 7)	(-3, 7)	(-3, -7)	(3, -7)
(4, 7)	(-4, 7)	(-4, -7)	(4, -7)
(5, 6)	(-5, 6)	(-5, -6)	(5, -6)
(6, 5)	(-6, 5)	(-6, -5)	(6, -5)
(7, 4)	(-7, 4)	(-7, -4)	(7, -4)
(7, 3)	(-7, 3)	(-7, -3)	(7, -3)
(8, 2)	(-8, 2)	(-8, -2)	(8, -2)
(8, 1)	(-8, 1)	(-8, -1)	(8, -1)
(8, 0)	(-8, 0)	(-8, 0)	(8, 0)

Q Determine the position of the circle in the first quadrant With radius 9 cm and the center of the circle is (2,2) by using midpoint Algm.

radius, $r = 9$ cm

$x_c = 2$, $y_c = 2$

$P_0 = 1 - r$; $1-9 = -8 < 0$

K	P_K	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	P_{k+1}
0	-8	(1, 9)	2	18	$P_{k+1} = P_k + 2x_{k+1} + 1$ $= -8 + 2 + 1 = -5$
1	-5	(2, 9)	4	18	$P_{k+1} = -5 + 4 + 1 = 0$
2	0	(3, 8)	6	16	$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$ $= 0 + 6 + 1 - 16 = -9$
3	-9	(4, 8)	8	16	$P_{k+1} = -9 + 8 + 1 = 0$
4	0	(5, 7)	10	14	$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$ $= 0 + 10 + 1 - 14$ $= -3$
5	-3	(6, 7)	12	14	$P_{k+1} = -3 + 12 + 1 = 10$
6	10	(7, 6)			

Coordinates (x,y) can be calculated as $x = x + x_c$, $y = y + y_c$

$Q_1(x, y)$	$Q_2(-x, y)$	$Q_3(-x, -y)$	$Q_4(x, -y)$
(2, 11)	(-2, 11)	(-2, -11)	(2, -11)
(3, 11)	(-3, 11)	(-3, -11)	(3, -11)
(4, 11)	(-4, 11)	(-4, -11)	(4, -11)
(5, 10)	(-5, 10)	(-5, -10)	(5, -10)
(6, 10)	(-6, 10)	(-6, -10)	(6, -10)
(7, 9)	(-7, 9)	(-7, -9)	(7, -9)
(8, 9)	(-8, 9)	(-8, -9)	(8, -9)
(9, 8)	(-9, 8)	(-9, -8)	(9, -8)
(9, 8)	(-9, 8)	(-9, -8)	(9, -8)
(9, 7)	(-9, 7)	(-9, -7)	(9, -7)
(10, 6)	(-10, 6)	(-10, -6)	(10, -6)
(10, 5)	(-10, 5)	(-10, -5)	(10, -5)
(11, 4)	(-11, 4)	(-11, -4)	(11, -4)
(11, 3)	(-11, 3)	(-11, -3)	(11, -3)
(11, 2)	(-11, 2)	(-11, -2)	(11, -2)

Bresenham's circle Drawing Algorithm

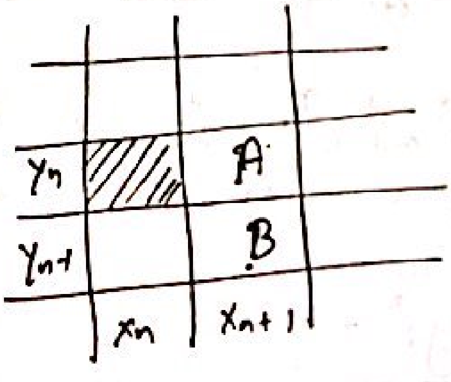
Bresenham's method of drawing the circle is an efficient method because it avoids the square root calculation of the mid point circle drawing by adopting only integer operation.

The bresenham's circle drawing algorithm considers the eight way symmetry of the circle. It plots 1/8 part of the circle from 90° to 45°. As the circle is drawn from 90° to 45°, the x-moves in +ve direction and y-moves in -ve direction

The new points closest to the true circle can be formed by applying two options

a) Increment in positive x direction by one unit or
b) Increment in positive x direction and negative y direction both by one unit

If the current point with coordinates (x_n, y_n) then the next point can be either (x_{n+1}, y_n) or (x_{n+1}, y_{n-1}).



The distance of pixel from A and B from the origin (0,0) are given by

$$d_A = \sqrt{(x_{n+1}-0)^2 + (y_n - 0)^2}$$

ie $d_A = \sqrt{x_{n+1}^2 + y_n^2}$

$d_B = \sqrt{x_{n+1}^2 + y_{n-1}^2}$

The distance of pixels A and B from the true circle whose radius 'r' are given as

$$\delta_A = d_A - r \text{ and } \delta_B = d_B - r$$

To avoid square root in derivation of decision variable we use $\delta_A = d_A^2 - r^2$ and $\delta_B = d_B^2 - r^2$

δ_A is always positive and δ_B is always negative. δ_A is always because distance from the origin to A is always greater than the radius of the circle and the distance from the origin to B is always ~~to~~ less than the actual radius of the circle.

∴ Decision parameter $p_k = \delta_A + \delta_B$

Let the point be (x_{i+1}, y_i) and (x_{i+1}, y_{i-1})

Then $p_k = x_{i+1}^2 + y_i^2 - r^2 + x_{i+1}^2 + y_{i-1}^2 - r^2$

Sub $x_{i+1} = x_i + 1$ and $y_{i-1} = y_i - 1$

$$= (x_i+1)^2 + y_i^2 + (x_i+1)^2 + (y_i-1)^2 - 2r^2$$

$$\boxed{p_k = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2}$$

$$P_{K+1} = 2(x_{i+1}+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$$

$$= 2(x_i+1+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$$

$$= 2(x_i+2)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$$

$$P_{K+1} - P_K = 2(x_i+2)^2 + y_{i+1}^2 + y_{i+1}^2 - 2y_{i+1} + 1 - 2r^2$$
$$- 2(x_i+1)^2 - y_i^2 - y_i^2 + 2y_i - 1 + 2r^2$$

$$= 2(x_i^2 + 4x_i + 4) + y_{i+1}^2 + y_{i+1}^2 - 2y_{i+1} + 1 - 2r^2$$
$$- 2(x_i^2 + 2x_i + 1) - y_i^2 - y_i^2 + 2y_i - 1 + 2r^2$$

$$= \cancel{2x_i^2} + \overset{4x_i}{8x_i} + \overset{6}{8} + 2y_{i+1}^2 - 2y_{i+1} + 1 - \cancel{2r^2}$$
$$- \cancel{2x_i^2} - 4x_i - 2 - 2y_i^2 + 2y_i - 1 + \cancel{2r^2}$$

$$P_{K+1} - P_K = 4x_i + 6 + 2y_{i+1}^2 - 2y_{i+1} - 2y_i^2 - 2y_i$$

$$\boxed{P_{K+1} = P_K + 4x_i + 2y_{i+1}^2 - 2y_{i+1} - 2y_i^2 + 2y_i + 6}$$

If $P_K < 0$ then x_{i+1}, y_i ie next $y_{i+1} = y_i$

$$\therefore P_{K+1} = P_K + 4x_i + 2y_i^2 - 2y_i - 2y_i^2 + 2y_i + 6$$

$$\boxed{P_{K+1} = P_K + 4x_i + 6 \quad \text{if } P_K < 0}$$

If $P_K > 0$ then x_{i+1}, y_{i-1} ie next $y_{i+1} = y_{i-1}$

$$P_{K+1} = P_K + 4x_i + 2(y_{i-1})^2 - 2y_{i-1} - 2y_i^2 + 2y_i + 6$$

sub $y_{i-1} = y_i - 1$

$$= P_K + 4x_i + 2(y_i - 1)^2 - 2(y_i - 1) - 2y_i^2 + 2y_i + 6$$

$= p_k + 4x_i + 2y_i^2 - 4y_i + 2 - \cancel{2y_i} + 2 - \cancel{2y_i^2} + \cancel{2y_{i+1}}$

$= p_k + 4x_i - 4y_i + 10$

$$\boxed{\therefore\ p_{k+1} = p_k + 4(x_i - y_i) + 10 \quad \text{if } p_k > 0}$$

Initial decision parameter p_0 can be calculated from the initial point $(x_0, y_0) = (0, r)$

ie $p_k = 2(x_i+1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$

Sub $(0, r)$ for x & y

$= 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2$

$= 2 + \cancel{r^2} + \cancel{r^2} - 2r + 1 - \cancel{2r^2}$

$= 3 - 2r$

$$\boxed{\therefore\ p_0 = 3 - 2r}$$

<u>Algorithm for Bresenham's circle drawing</u>

Step 1 : Read the radius 'r' of the circle and center (x_c, y_c)

Step 2 : Calculate the initial value of the decision parameter as $p_0 = 3 - 2r$ and initial point as $(x_0, y_0) = (0, r)$

Step 3 : At each x_k position starting at $k = 0$. perform the following test.

If $p_k < 0$ then next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and

$p_{k+1} = p_k + 4x_k + 6$

otherwise the next point on the circle is $(x_{k+1}, y_k - 1)$ and $p_{k+1} = p_k + 4(x_k - y_k) + 10$

Step 4: Determine symmetry points of other seven octants.

Step 5: Move each calculated pixel position (x, y) onto the circular path centred on (x_c, y_c) and plot the coordinate values.

$$x = x + x_c, \quad y = y + y_c$$

Step 6: Repeat step 3 to 5 until $x \geq y$.

```
Procedure Bresenham's circle { xcenter, ycenter, Radius : integer);
Var
  p, x, y : integer;
  procedure plotpoints;
  begin
     Setpixel (xcenter + x, ycenter + y , 1);
     Set pixel ( xcenter -x , ycenter + y , 1);
     Set pixel (xcenter + x, ycenter - y, 1);
     Set pixel (xcenter -x, ycenter -y , 1);
     Set pixel (xcenter + y, ycenter +x, 1);
     Set pixel (xcenter -y, ycenter +x, 1);
     Set pixel ( xcenter +y , ycenter -x ,1);
     Set pixel (xcenter -y , ycenter -x , 1);
end; plot points
 begin
     x:=0;
     y:= radius;
     plotpoints ;
     p:=3 -2r
```

```
while x < y do
    begin
        if p < 0 then
            x := x + 1
        else
            begin
                x = x + 1
                y = y - 1
            end;
        if p < 0 then
            p = p + 4x + 6
        else
            p = p + 4(x - y) + 10
        . plot points
    end;
end; {Bresenhams circle}
```

Q Determine the positions on the circle having $r = 8$ and having centre position as $(x_c, y_c) = (30, 40)$ by using Bresenhams' drawing Algorithm.

Initial point $(x_0, y_0) = (0, 8)$

$P_0 = 3 - 2r = 3 - 16 = -13$

x	y	p
		-13
0	8	$p = P_k + 4x + 6 = -13 + 4 + 6 = -3 < 0$
1	8	$p = p_k + 4x + 6 = -3 + 8 + 6 = 11 > 0$
2	8	$p = p_k + 4(x-y) + 10 = 11 + 4(-4) + 10 = 11 - 16 + 10 = 5 > 0$
3	7	$p = p_k + 4(x-y) + 10 = 5 + 4(-2) + 10 = 7 > 0$
4	6	
5	5	

$(x, y) = (x + x_c), (y + y_c)$

$x_c = 30$, $y_c = 40$

(x, y)
(30, 48)
(31, 48)
(32, 48)
(33, 47)
(34, 46)
(35, 45)

Q Determine the positions on the circle having radius $r = 10$ and having centre as origin (0, 0) by using Bresenham's drawing Algorithms.

$(x_0, y_0) = (0, 10)$

$p_0 = 3 - 2 \times 10 = -17$

p.

X	Y
0	10
1	10
2	10
3	9
4	9
5	8
6	7
7	6

$p_0 = 3 - 20 = -17 < 0$

$p = p + 4x + 6 = -17 + 4 + 6 = -7 < 0$

$p = p + 4x + 6 = -7 + 4 \times 2 + 6 = 7 > 0$

$p = p + 4(x - y) + 10 = 7 + 4(-6) + 10 = -7 < 0$

$p = p + 4x + 6 = -7 + 16 + 6 = 15 > 0$

$p = 15 + 4(-3) + 10 = 13 > 0$

$p = 13 + 4(-1) + 10 = 19 > 0$

Filled Area Primitives

A standard output primitive in general graphics package is a solid color or patterned polygon area.

There are two basic approaches to fill the area on the raster system.

i) One way to fill an area is to determine the overlap intervals for scan lines that cross the area.

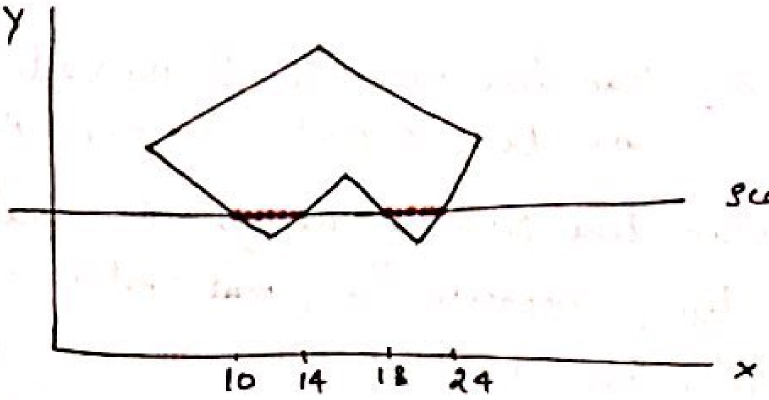
ii) Another method for area filling is to start from a given interior position and paint outward from this point until we a specified boundary condition is encountered.

There are mainly 3 kinds of polygon filling algorithm.

i) Scan line polygon Fill Algorithm
ii) Boundary Fill Algorithm
iii) Flood Fill Algorithm.

Scan line polygon Fill Algorithm

The following figure illustrates the scan-line procedure for solid filling of polygon areas. For each scan line crossing a polygon, the area fill algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are sorted from left to right and the corresponding frame-buffer positions between each intersection pair are set to the specified fill color.

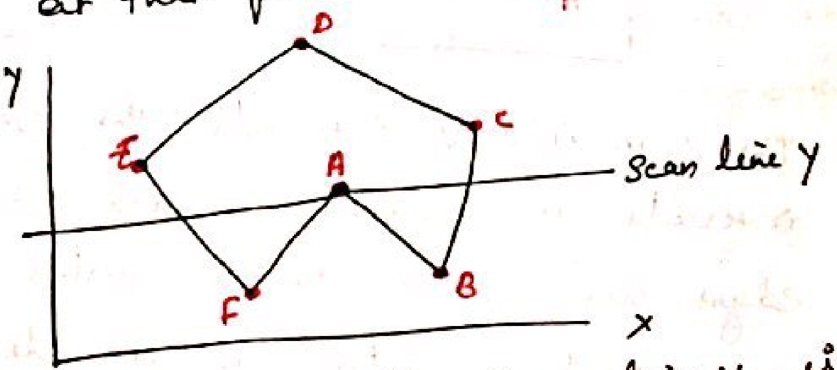


In the above examples, Four pixel intersections position with the polygon boundaries define the two stretches of interior pixels from x=10 to x=14 & x=18 to x=24

Special cases in scan line intersections with polygon:-

Case 1: Scan line pass through the vertices

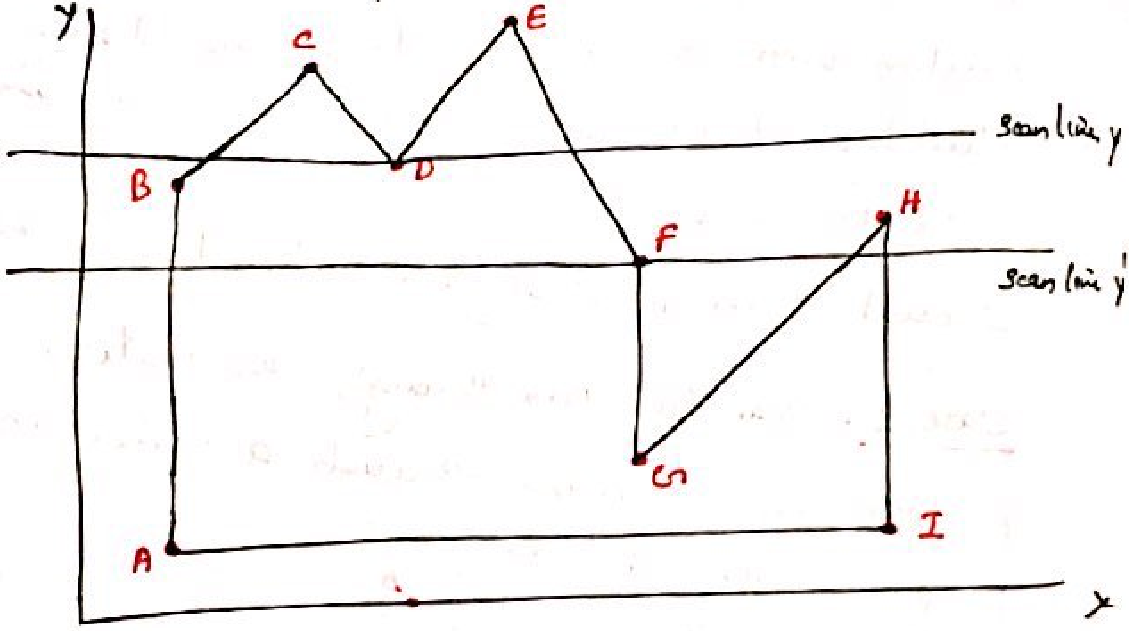
A scan line passing through a vertex intersects two polygon edges at that position.



In the above given figure the scan line Y is passing through the vertex 'A'. which intersects two edges of vertex 'AB' and 'AF'. In this case the intersections A with scan line Y will be considered as two and count the single intersection point as two.

Case 2 : Scan line pass through the vertex whose edges are both opposite to each other

A scan line passing ~~ing~~ through the vertex whose edges are lying opposite to that vertex. This case requires special processing. The following figure show the case 2 intersection of vertex with scan line



The scan line 'y'' intersects the vertex 'F' which intersects two edges 'GF' and 'EF'. Both of the edges are opposite to the vertex F.

These kind of vertex can be identified by tracing the polygon boundary either in clockwise or anticlockwise direction. and observe the relative change in the y-cordinate value in vertex as we move from one edge to the next.

Difference between the intersection of the scanline y and y' with the vertex is that.

For scanline y, the edges of the intersection vertex 'D' are on the same side of the scanline, i.e. it is above the scanline

For scanline y', the edges intersecting at the vertex 'F' are on either sides of the vertex intersection with scanline.

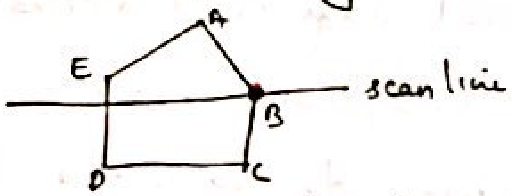
Vertex counting in a scanline :-

i) Traverse along the polygon boundary clockwise or anticlockwise

ii) Observe the relative change in y-value of the edges on either side of the vertex (as we move from one edge to another).

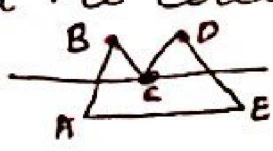
iii) check the conditions.

a) If the y-coordinate value of the two edges in the intersecting vertex are monotonically increase or decrease then count the intersected vertex as a single intersection point for the scan line passing through it.



Count 'B' as one.

b) Else if the y-coordinate value of the shared vertex represents the local minimum or local maximum on the polygon boundary. Increment the intersection count. i.e. count the intersection point as two.



Count the intersection point 'C' as two

Implementation of above cases.

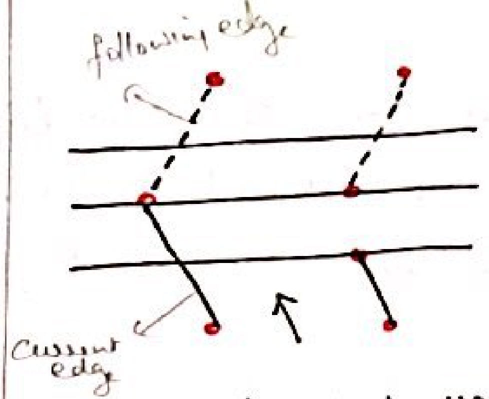
To resolve the question whether we should count a vertex as one intersection or two ?

We have to shorten some polygon edges to split those vertices that should be counted as one intersection.

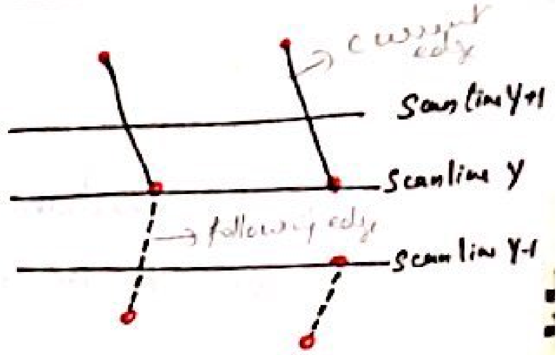
The nonhorizontal edges around the polygon boundary can be process either clock wise or anticlockwise and determine whether the edge and the next horizontal edge is monotonically increase or decrease the y-coordinate value. If the y-value is increasing or decreasing then the lower ~~or~~ upper edge can be shortened to ensure that only one intersection point is generated for the scan line going through the common vertex joining two edges. The following figure illustrates shortening of edge. ~~when the~~.

(a) when the endpoint y coordinate of the two edges are increasing, the y value of the upper endpoint for the current edge is decreased by 1.

(b) when the endpoint y value are monotonically decreasing, we decrease the y-coordinate of the upper endpoint of the edge which follows the current edge.



Y coordinate of the upper endpoint of current edge is decreased by 1



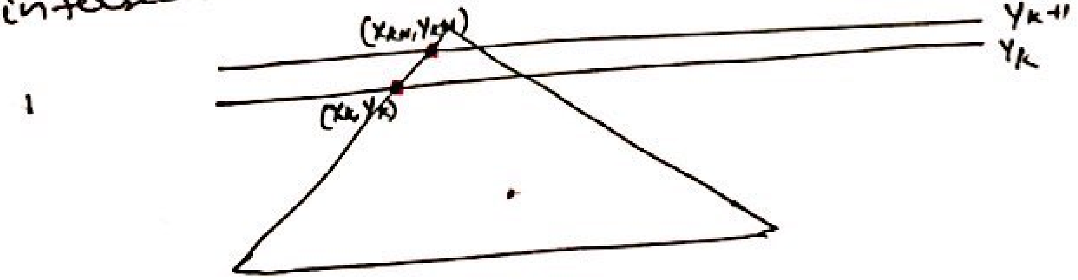
Y coordinate of the upper endpoint of the next edge is decreased by 1

Two important feature of scanline based polygon filling Algm

(i) Scanline Coherence - Values do not change much from one scanline to the next - the coverage of a face one scanline typically differs from the previous one scanline

(ii) Edge coherence - edges intersected by Scanline 'i' are typically intersected by scanline i+1.

Following figure shows two successive scan line crossing a left edge of a polygon. The slope of this polygon boundary line can be expressed in terms of the scan line intersection coordinates



$$\text{Slope } m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

The change in y coordinate between the two scan line is $y_{k+1} - y_k = 1$

The x-intersection value x_{k+1} on the upper scan line can be determined from the x- intersection value x_k on the preceding scan line as (By DDA Algm)

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

Along the edge with slope m, the intersection x_k value for the scan line k above the initial scan line can be calculated as

$$x_k = x_0 + \frac{k}{m}$$

Thus the incremental calculation of x intercepts along an edge for sucessive scan line can be expressed as $$x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$$ $\left| m = \frac{\Delta y}{\Delta x} \right.$

Inside-Outside Test

Area filling algorithms need to identify the interior region of the objects.

In elementary geometry the polygon is usually defined as no self intersection. The edges of the standard-polygon are joined only at the vertices, or the edges have no common endpoints in the plane.

It is difficult to find the interior region of the polygon whose edges are ~~intersections~~ intersecting in the plane.

Inorder to find the interior and exterior regions of such shapes graphics package normally use two method.

i) odd-even rule
ii) Non-zero Winding Number

odd-even rule (odd parity)

In this method a line is drawn from any position 'p' to a distant point outside the coordinate extents of the object and counting the number of edges crossing along the line. If the number of polygon edges crossed by this line is odd, then p is an interior point otherwise if the number of polygon edges crossed by the line is even then p is an exterior point.

NB:-

To obtain an accurate edge count, we must be sure that the line path we choose does not intersect any polygon vertices.

The following figure shows the interior and exterior regions obtained from the odd-even rule for a self-intersecting set of edges.

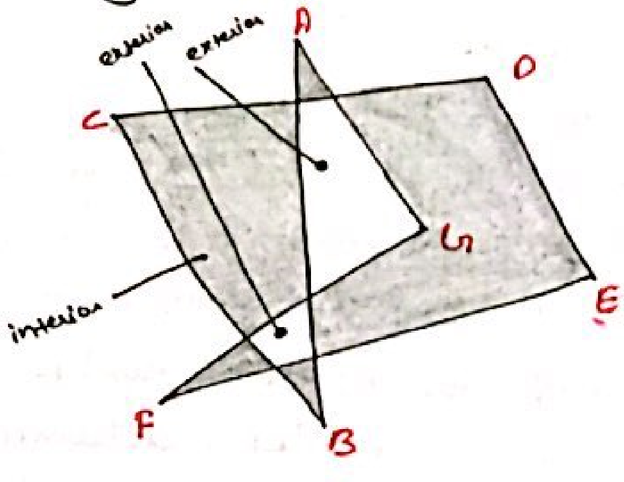


fig: odd even Rule.

Non Zero Winding Number Rule

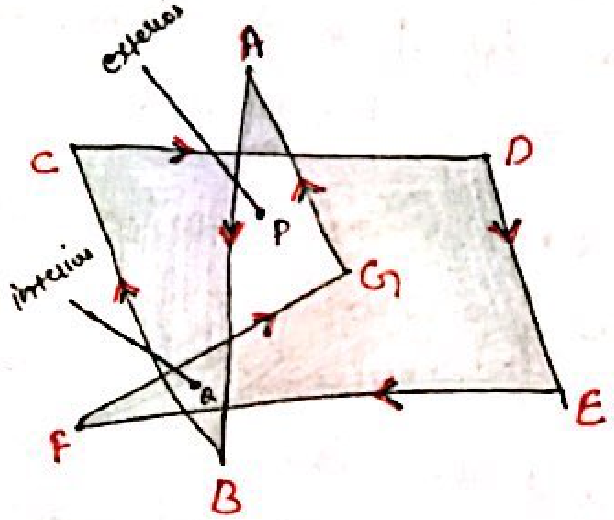
This method counts the number of times the polygon edges wind around a particular point in the ~~clockwise~~ counterclockwise direction. This count is called the winding number. The interior points of a two-dimensional object are defined to be those that have a nonzero value for the winding number.

The non zero winding number rule is applied to the polygon by initializing the winding number to 0 and a line is drawn from any position 'p' to a distant point beyond the coordinate extents to the object. The drawn line must not pass through any vertices. The no: of edges that cross the drawn line is counted as we move along the line from position 'p' to a distant point beyond the coordinate extents of the object. The winding number of each edge is taken depend on the direction of the edges and the value of the direction is as follows.

i) An edges cross the line from right to left is 1
ii) An edge cross the line from left to right is -1
iii) An edge cross the line from top to bottom is 1
iv) An edge cross the line from bottom to top is -1

The final value of the winding number, after all edge crossings have been counted, determines the relative position of 'p'. If the winding number is nonzero 'p' is defined to be an interior point, otherwise 'p' is taken to be the exterior point.

The following figure shows the interior and exterior region defined by the nonzero winding for a self-intersecting set of edges.



P = AB + CD
1 - 1 = 0

Q = BC + FG
-1 - 1 = -2 != 0

Boundary Fill Algorithm

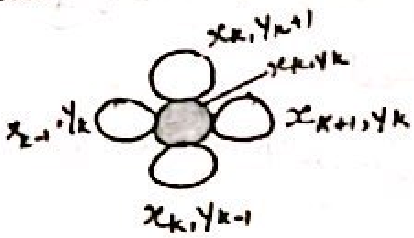
Boundary Fill Algorithm fill the interior Region of the boundary with one color by comparing the color in the boundary area.

This method start at a point inside a Region and paint the interior outward toward the boundary. If the boundary is Specified in a single color, the fill algorithm proceeds outward pixel by pixel until the boundary color is encountered. This method is called boundary fill algorithm and this method is useful in painting packages, where the interior points are easily selected.

Boundary Fill Algorithm uses two methods for filling the neighbouring pixels with color.

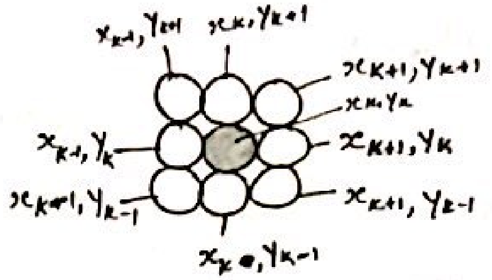
i) 4- Connected
ii) 8- Connected

In the 4-connected method, 4 neighbouring points are tested. These are the pixel positions that are right, left, above and below the current pixel.



8-Connected Method is used to fill more complex figures. This method includes the four diagonal pixels in the set of neighbouring position to be tested.

Scanned with CamScanner



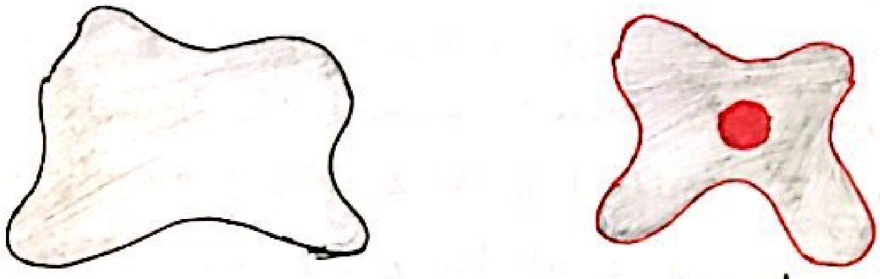


fig: Example color boundaries for a boundary-fill procedure

A boundary fill procedure accepts as input the coordinates of an interior point (x,y), a fill color, and a boundary color. Starting from (x,y) the procedure tests neighbouring positions to determine whether they are of the boundary color. If the neighbouring pixels tested are not fill with the the boundary color then they are painted with the fill colour and again their neighbours are tested using the algorithm. This process continues until all pixels up to the boundary color for the area have been tested.

```
procedure boundaryFill 4 (x, y, fill, boundary : integer);
var
   current : integer;
begin
   current = getpixel (x, y);
   if (current <> boundary) and (current <> fill) then
      begin
      set pixel (x, y, fill);
```

boundaryfill 4 (x+1, y, fill, boundary);
boundaryfill 4 (x-1, y, fill, boundary);
boundaryfill 4 (x, y+1, fill, boundary);
boundaryfill 4 (x, y-1, fill, boundary)

end

end; { boundaryfill }

Flood Fill Algorithm

Flood fill Algorithm is used to fill (recolor) an area that is not defined within a single color boundary. Such area can be paint by replacing a specified interior color instead of searching for a boundary color value.

In flood fill Algorithm we start from a speicfied interior point (X, Y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

if the area we want to paint has more than one interior color, first reassign pixel values so that all interior points have the same color.

4-connected or 8-connected approach can be used to step the pixel positions until all the interior points have been painted.

following procedure Flood fills a 4-Connected region recursively, starting from the input positions.

3/11/19

```
Procedure flood Fill 4 (x, y, fill color, old color : integer);
begin
  If getpixel (x, y) = oldcolor then
    begin
      setpixel (x, y, fillcolor);
      floodfill 4 (x+1, y, fillcolor, oldcolor);
      floodfill4 (x-1, y, fill color, old color);
      flood fill 4 (x, y+1, fillcolor, oldcolor);
      floodfill 4 (x, y-1, fill color, old color);
    end
end; {flood fill }
```

Two Dimensional Transformations

Changes in orientation, size and shape are accomplished with geometric transformations that alter the coordinate descriptions of the objects. The basic geometric transformations are

i) Translation
ii) Rotation
iii) Scaling

Other form of Transformation applied to the objects are

i) Reflection
ii) Shear

Basic Transformation

These transformation are used to reposition and resize two dimensional objects.

Translation

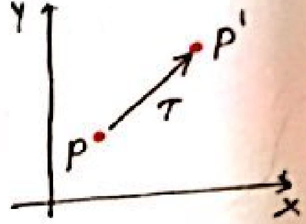
A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another

A 2-D point is translate by adding translation distances, t_x and t_y to the original coordinate position (x, y) to move the point to a new position (x', y')

$$\boxed{\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}} \longrightarrow (1)$$

The translation distance pair (t_x, t_y) is called a translation vector or shift vector

The above translation equation can be expressed as a single matrix equation by using column vectors to represent coordinate positions and the translation vector



$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \underset{(P)}{\begin{bmatrix} x \\ y \end{bmatrix}} + \underset{(T)}{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}$$

The 2-D translation equation in the matrix can be in the form $P' = P + T \longrightarrow$ ②

The translation matrix equations can be expressed in terms of coordinate row vectors instead of column vectors

The column vector can be represented as $P = [x \ y]$ and $T = [t_x, t_y]$ ie $[x' \ y'] = [x \ \ y] + [t_x \ \ t_y]$

Translation is a rigid-body transformation that moves objects without deformation. ie every point on the object is translated by the same amount.

* A straight line segment is translated by applying the transformation equation ② to each of the line endpoints and redraw the line between the new endpoint positions.
* Polygons are translated by adding the translation vector to the coordinate positions of each vertex and regenerating the polygon using the new set of vertex coordinates and the current attribute settings

Eg:- Consider the triangle polygon with the three vertex (9,2), (15,5) and (20,2) and the translation vector be the (-5.50, 3.75)

$(x_1, y_1) = (9, 2)$ $\qquad S_x = -5.50$

$(x_2, y_2) = (15, 5)$ $\qquad S_y = 3.75$

$(x_3, y_3) = (20, 2)$

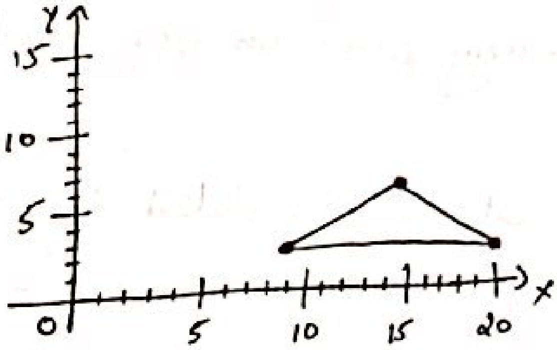
$x_1' = x_1 + S_x = \cancel{9+5.50}\ 9 + -5.50 = 3.5$

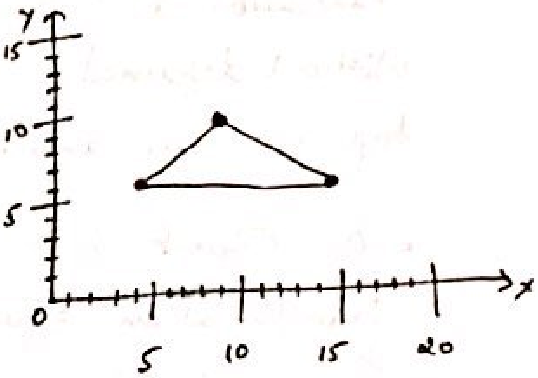
$y_1' = y_1 + S_y = 2 + 3.75 = 5.75$

$(x_1', y_1') = (3.5, 5.75)$

$x_2' = x_2 + S_x = 15 + -5.50 = 9.5$
$y_2' = y_2 + S_y = 5 + 3.75 = 8.75$ | $(x_2', y_2') = (9.5, 8.75)$

$x_3' = x_3 + S_x = 20 + -5.50 = 14.5$
$y_3' = y_3 + S_y = 2 + 3.75 = 5.75$ | $x_3', y_3' = (14.5, 5.75)$





<u>Rotation</u>

A 2-D rotation is applied to an object by repositioning it along a ~~circle~~ circular path in the xy plane.

Rotation is generated by specify a rotation angle θ and the position (x_r, y_r) of the <u>rotation point (or pivot point)</u> about which the object is to be rotated.

positive values for the rotation angle defines · Counterclockwise rotation about the pivot point and the negative values rotate the object in the clockwise direction.

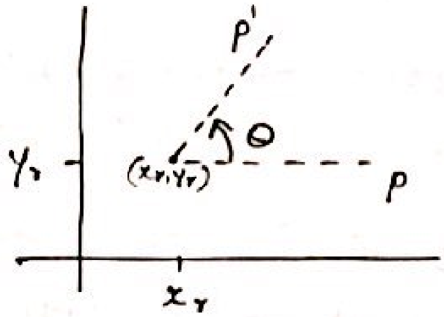
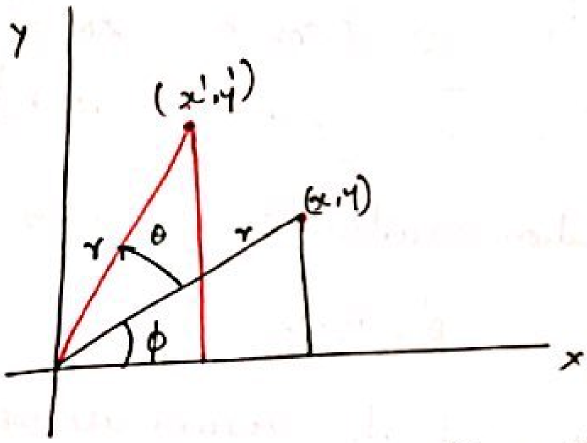


fig: Rotation of an object through angle θ about the pivot point

i) Rotation of a point at the <u>coordinate origin</u>:-

The angular and coordinate relationship of the original point and the transformed point position are shown in the following figure.



In the above figure 'r' is the radius which is constant distant of the point from the origin and the 'ϕ' is the original angular position of the point from the horizontal x-axis. and θ is the rotation angle. Using the trigonometric identities the original coordinates of the point can be expressed as

$$x = r\cos\phi \quad \longrightarrow ①$$
$$y = r\sin\phi$$

The transformed coordinates can be expressed interms of angle θ and ϕ as

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \quad \longrightarrow (2)$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

Substituting (1) in (2). The transformation equations for rotating a point at position (x,y) through an angle θ about the origin:

$$x' = x\cos\theta - y\sin\theta \qquad \text{ie} \quad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$
$$y' = x\sin\theta + y\cos\theta$$

The Rotation column matrix can be written as

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

The Rotation equation can be given as

$$P' = R \cdot P$$

When the coordinate positions are represented as row vectors instead of column vectors, the matrix product in rotation is transposed so that the transformed row coordinate vector $[x', y']$ is calculated as

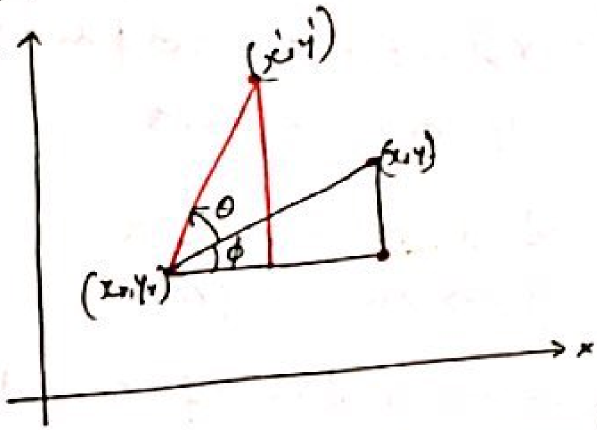
$$P'^T = (R \cdot P)^T$$
$$= P^T \cdot R^T$$

where $P^T = [x, y]$ and the Transpose R^T of matrix is obtained by interchanging rows & columns. For a rotation matrix transpose is obtained by simply changing the sign of sine terms

$$[x' \; y'] = [x \; y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

ii) Rotation of a point at the pivot position :-

Rotation of a point about an pivot position is illustrated in the following figure.



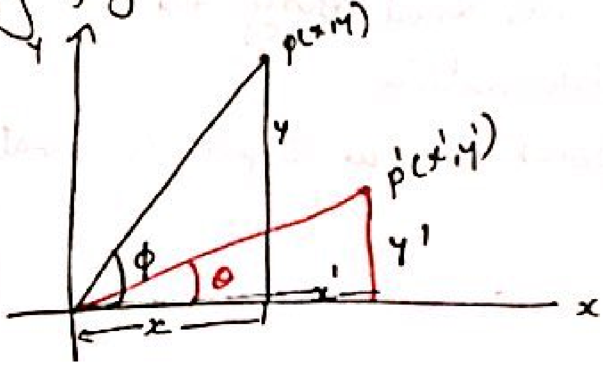
Using trigonometric relationship, the transformation equation for rotation of a point about any specified rotation position (x_r, y_r) is given by

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

iii) Rotation of a point in clockwise direction :-

The angular and the coordinate relationship of the original point and the transformed point position are shown in the following figure

old angle = ϕ
new angle of $p \to p' = (\phi - \theta)$



$x' = r\cos(\phi - \theta)$ $\qquad \because \cos\phi - \theta = \frac{x'}{r}$

$y' = r\sin(\phi - \theta)$ $\qquad \because \sin\phi - \theta = \frac{y'}{r}$

$x = r\cos\phi$

$y = r\sin\phi$

$$x' = r(\cos\phi\cos\theta + r\sin\phi\sin\theta)$$
$$= x\cos\theta + y\sin\theta$$

$$y' = r(\sin\phi\cos\theta - \cos\phi\sin\theta)$$
$$= y\cos\theta - x\sin\theta.$$

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

The column matrix of the above equation can be represented as

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

The row matrix of the rotation of object in clockwise direction is given by (Reverse the row & column of matrix R)

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

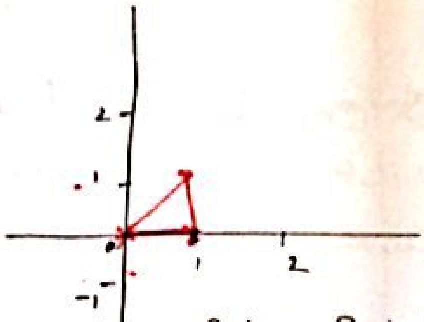
* Rotation are rigid body transformation that move objects without deformation.
* Every point on an object is rotated through the same angle.

* polygon are rotated by displacing each vertex through the specified rotation angle and regenerate the polygon using new vertices

Example:-

Q. Consider a triangle with coordinate (0,0) (1,0) (1,1) and the rotation angle θ is 90° (Anticlockwise). Find the new coordinates after the transformation

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



Before Rotation.

$\cos 90 = 0$

$\sin 90 = 1$

For coordinate (0,0)

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

$$= \begin{bmatrix} 0+0 \\ 0+0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$(0,0) \Longrightarrow (0,0)$

For (1,0)

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

$$= \begin{bmatrix} 0+0 \\ 1+0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$(1,0) \Longrightarrow (0,1)$

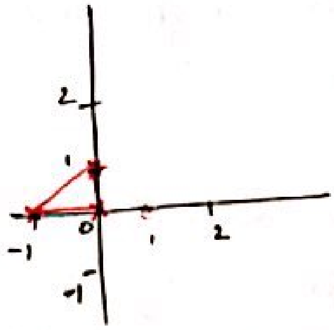
For (1,1)

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

$(1,1) \Longrightarrow (-1,1)$

$$= \begin{bmatrix} 0+-1 \\ 1+0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

The new transformed coordinates are (0,0) (0,1) (-1,1)



After Rotation

Q Consider the triangle with coordinates (0,0) (1,0), (1,1) and the rotation angle is 90° (clockwise). Find the new transformed coordinates

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

$\cos 90 = 0$

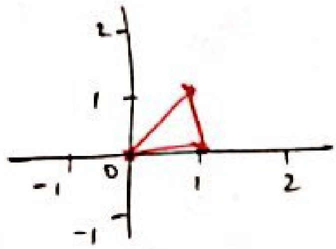
$\sin 90 = 1$

For (0,0)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0+0 \\ 0+0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$(0,0) \Rightarrow (0,0)$



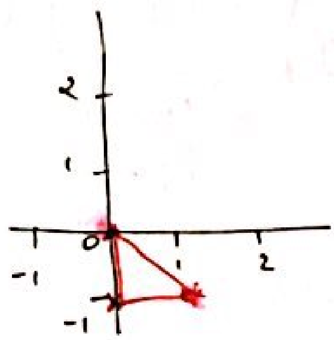
Before Rotation

For (1,0)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0+0 \\ -1+0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$(1,0) \Rightarrow (0,-1)$



After Rotation

For (1,1)

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

$$= \begin{bmatrix} 0+1 \\ -1+0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$(1,1) \Rightarrow (1,-1)$

Scaling

Scaling is a transformation that alters the size of an object. This operation is carried out by multiplying the coordinate values (x,y) of each vertex by scaling factors S_x and S_y to produce the transformed coordinates (x', y'). The scaling equation is given as below

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

Scaling factor S_x scales the object in x-direction

Scaling factor S_y scales the object in y-direction.

The above transformation equation can be written as a matrix form as given below

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

or

$$P' = S \cdot P$$

The size of the scaled object depends on the value of S_x and S_y.

i) If S_x and S_y are assigned any positive value between 0 & 1 reduces the size of the object and the transformation point is closer to the origin

ii) If the value of S_x and S_y are greater than 1 then the transformed points are away from the origin and the size of the object gets increased

iii) If S_x and S_y equals to one then the size of the object is unchanged.

iv) If the value S_x and S_x are same, then scaling will be done uniformly in both x and y axis.

v) Unequal values of S_x and S_y results in differential scaling.

NB:-

Objects transformed with the scaling equations are both scaled and repositioned.

Scaling can be performed with respect to the pivot point (x_f, y_f) or fixed point which remains unchanged after the scaling transformation. For a vertex with coordinates (x, y), the scaled coordinates (x', y') with respect to fixed point (x_f, y_f) are calculated as

$$x' = x_f + (x - x_f) S_x$$
$$y' = y_f + (y - y_f) S_y$$

* polygon are scaled by applying transformation to each vertex and then regenerating the polygon using the transforming vertices.

Example:

Q. Consider the square with coordinates (0,0) (2,0) (0,2) and (2,2) and the scaling factor $S_x = 2$ and $S_y = 3$. Find the new coordinates in the scaling transformation

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

for (0,0)

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

$$= \begin{bmatrix} 0+0 \\ 0+0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

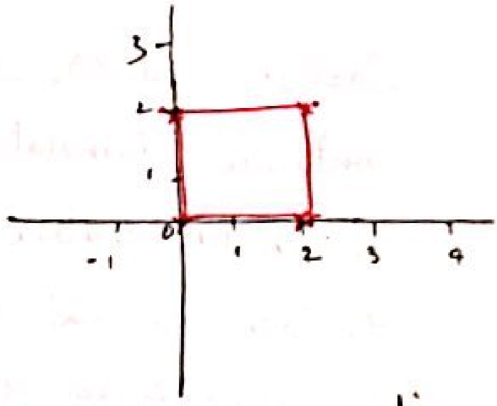
$(0,0) \Rightarrow (0,0)$

For $(2,0)$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4+0 \\ 0+0 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$(2,0) \Rightarrow (4,0)$



Before scaling

For $(0,2)$

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

$$= \begin{bmatrix} 0+0 \\ 0+6 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \end{bmatrix}$$

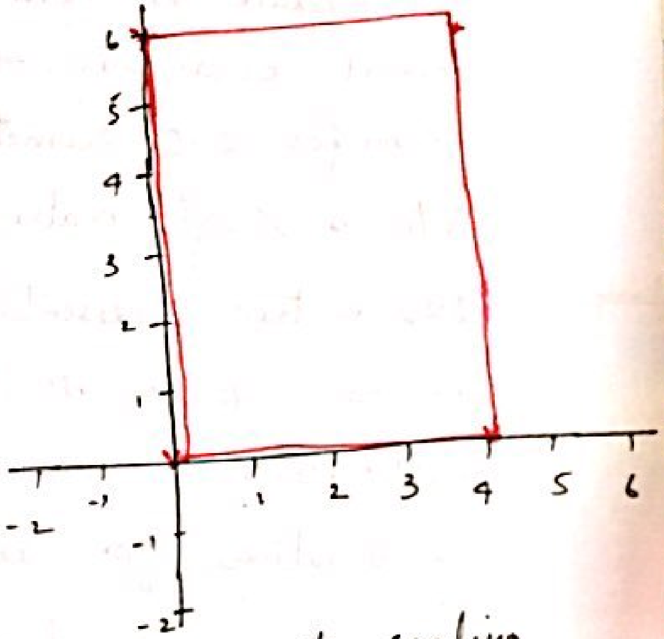
$(0,2) \Rightarrow (0,6)$

For $(2,2)$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 4+0 \\ 0+6 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

$(2,2) \Rightarrow (4,6)$



After scaling

Homogenous Coordinate System

The basic transformation can be expressed in the general matrix form

$$P' = M_1 \cdot P + M_2$$

where P' and P can be represented as column vector Matrix M_1 is a 2×2 array contain Multiplicative

Scanned with CamScanner

factors and M_2 is a two-element column matrix containing translational terms. For translation, M_1 is the identity matrix. For rotation or scaling, M_2 contains the translational terms associated with the pivot point or scaling fixed point.

To produce the sequence of transformations directly from initial coordinates, the multiplicative and translational terms for 2-D geometric transformations can be combined into a single matrix representation by expanding 2x2 matrix representations by 3x3 matrices. This allows us to express all transformation equations as matrix multiplication providing by the expansion of matrix representations for coordinate positions.

Each cartesian coordinate (x, y) of the 2-D geometric transformation can be expressed as the ~~co~~ Homogenous coordinate triple (x_h, y_h, h) where

$$x = \frac{x_h}{h} \quad , \quad y = \frac{y_h}{h}$$

'h' can be any nonzero value for the 2-D geometric transformation. For convenience the value of h is simply set as 1 ie $h = 1$. Each 2-D position is then represented with homogenous coordinates $(x, y, 1)$

The Homogenous matrix representation for translation can be given as

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

$x' = x + t_x$
$y' = y + t_y$

which can be written as $p' = T(t_x, t_y) \cdot p$

The inverse of the translation matrix is obtained by replacing the translation parameters t_x and t_y with their negatives $-t_x$ and $-t_y$.

Homogenous matrix representation of the rotation transformation about the coordinate origin can be written as

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

and can be represented as $p' = R(\theta) \cdot p$

The inverse rotation matrix can be represented when θ is replaced with $-\theta$.

Homogenous matrix representation of the scaling transformation relative to the coordinate origin is expressed as

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

and can be expressed as $p' = S(S_x, S_y) \cdot p$

The inverse scaling matrix can be obtained by replacing the scaling parameters S_x and S_y with ($1/S_x$ and $1/S_y$).

Matrix Formulation and Concatenation of Transformations

Matrix can be set up for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transformations. Products of transformation matrix is referred as a concatenation or composite of matrices.

For column matrix representation of coordinate position, composite transformation are formed by multiplying matrices in order from right to left. i.e. each successive transformation matrix premultiplies the product of the preceding transformation matrices.

Translation

If two successive translation vectors (tx_1, ty_1) and (tx_2, ty_2) are applied to a coordinate position P: the final transformed location P' is calculated as

$$P' = T(tx_2, ty_2) \cdot \{T(tx_1, ty_1) \cdot P\}$$
$$= \{T(tx_2, ty_2) \cdot T(tx_1, ty_1)\} \cdot P$$

where P and P' are the homogenous coordinate column vectors.

The composite transformation matrix for this sequence of translation is

$$\begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x_1} + t_{x_2} \\ 0 & 1 & t_{y_1} + t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$T(t_{x_2}, t_{y_2}) \cdot T(t_{x_1}, t_{y_1}) = T(t_{x_1} + t_{x_2}, t_{y_1} + t_{y_2})$$

* Two successive translation are additive

Rotations

Two successive rotations applied to point P produce the transformed position

$$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\}$$
$$= \{R(\theta_2) \cdot R(\theta_1)\} \cdot P$$

* Two successive rotation matrix are additive.

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2 & -\cos\theta_1\sin\theta_2 - \sin\theta_1\cos\theta_2 & 0 \\ \sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2 & -\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ie $R(\theta_2) \cdot R(\theta_1) = R(\theta_1 + \theta_2)$

Final rotated coordinates can be calculated with the composite rotation matrix as

$$P' = R(\theta_1 + \theta_2) \cdot P$$

Scalings

Concatenating transformation matrices of two successive scaling operations produces the following composite scaling matrix

$$\begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{x_1} \cdot S_{x_2} & 0 & 0 \\ 0 & S_{y_1} \cdot S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$S(S_{x_2}, S_{y_2}) \cdot S(S_{x_1}, S_{y_1}) = S(S_{x_1} \cdot S_{x_2}, S_{y_1} \cdot S_{y_2})$$

* Two successive scaling operation is multiplicative.

General pivot point Rotation

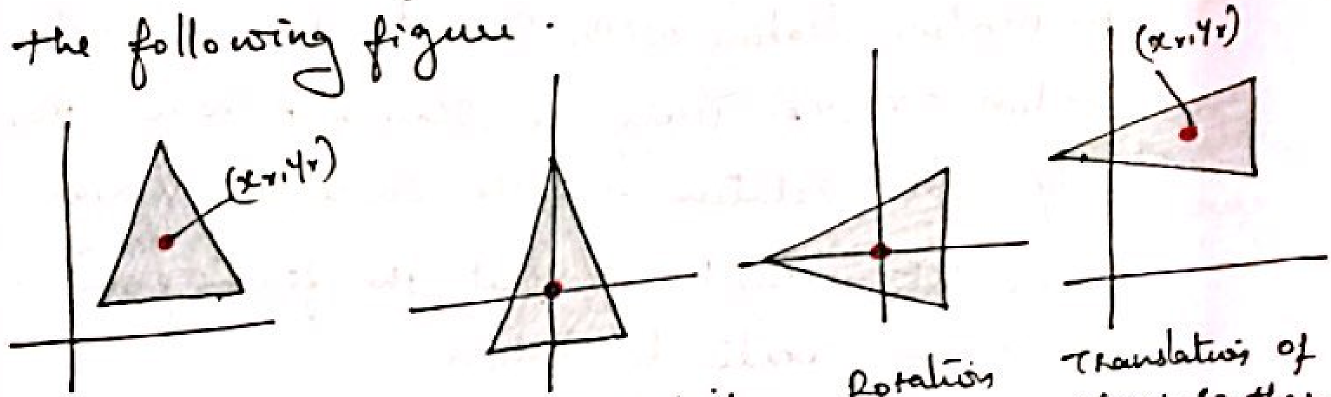
Graphics package provides rotation function for revolving object about the coordinate origin.

If the rotation needs to generate about any selected pivot point (x_r, y_r) then the following sequence of operations need to be performed.

Scanned with CamScanner

1. Translate the object so that the pivot point position is moved to the coordinate origin
2. Rotate the object about the coordinate origin
3. Translate the object so that the pivot point is returned to its original position.

The above transformation sequence is illustrated in the following figure.



The composite transformation matrix for the above sequence is obtained with the concatenation

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

The above matrix can be expressed in the form

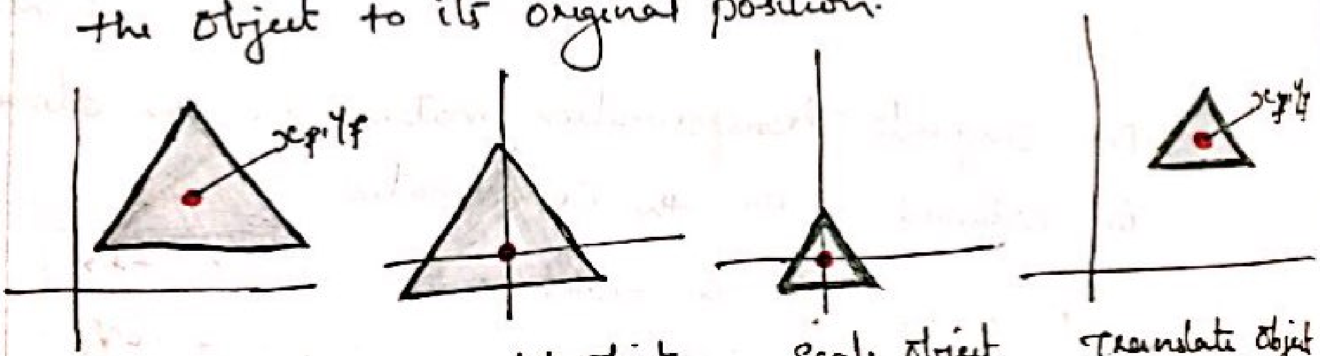
$$T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r) = R(x_r, y_r, \theta)$$

where $T(-x_r, -y_r) = T^{-1}(x_r, y_r)$.

General Fixed-point scaling

The following figure illustrates a transformation sequence to produce scaling with respect to a selected fixed position (x_f, y_f) using a scaling function that can only scale relative to the coordinate origin.

1. Translate object so that the fixed point coincides with the coordinate origin
2. Scale the object with respect to the coordinate origin
3. Use the inverse translation of step 1 to return the object to its original position.



Original position of object and fixed point | Translate object so that fixed point (x_f, y_f) is at origin | Scale object with respect to origin | Translate object so that the fixed point is returned to the position (x_f, y_f)

Concatenating the matrices for these three operations produces the required scaling matrix.

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

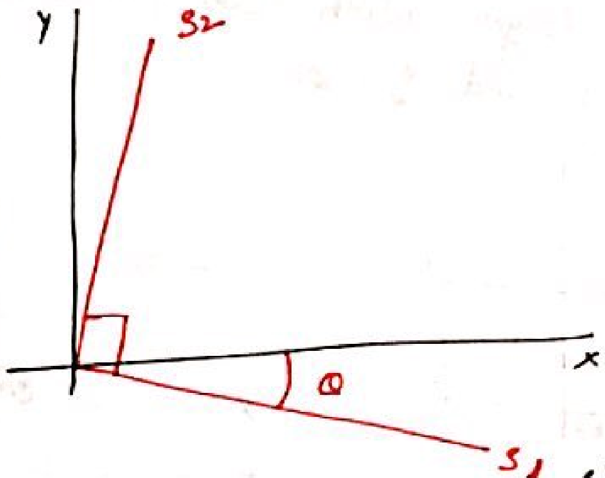
or

$$T(x_f, y_f) \cdot S(s_x, s_y) \cdot T(-x_f, -y_f) = S(x_f, y_f, s_x, s_y)$$

General scaling Directions

Parameters S_x and S_y scale objects along the x and y directions. The object can be scaled in other directions by rotating the object to align the desired scaling directions with the coordinate axes before applying the scaling Transformations

A scaling factors with values specified by parameters S_1 and S_2 in the direction shown in the following figure.



Following are the sequence of steps that should be followed for accomplishing scaling without changing the orientation of the object.

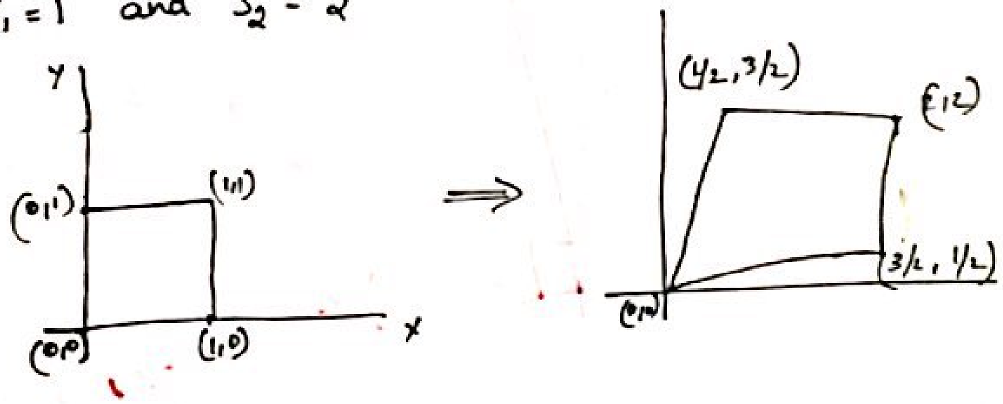
1. perform rotation so that the directions for S_1 and S_2 coincide with the x and y axes.
2. Scale the object
3. perform opposite rotation to return points to their original orientations

The composite matrix resulting from the product of these three transformation is

$$R^{-1}(\theta) \cdot S(s_1, s_2) \cdot R(\theta)$$

$$= \begin{bmatrix} s_1\cos^2\theta + s_2\sin^2\theta & (s_2 - s_1)\cos\theta\sin\theta & 0 \\ (s_2 - s_1)\cos\theta\sin\theta & s_1\sin^2\theta + s_2\cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Eg: Unit square can be transformed into parallelogram by stretching it along the diagonal from (0,0) to (1,1). Rotate the diagonal onto the y axis and double its length with transformation parameters $\theta = 45°$, $S_1 = 1$ and $S_2 = 2$



Concatenation properties

* Matrix multiplication is associative.

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

* Matrix product can be evaluated either from left to right or from right to left.

* Transformation products may not be commutative.
  ie $A \cdot B \neq B \cdot A$

Scanned with CamScanner

* The order is important if we are translate and Rotate an object.

(8) For special cases sequence of transformation of the same kind, the multiplication of transformation matrices is commutative.

General Composite Transformations & Computational Efficiency

A general 2-D transformation, representing a combination of translations, rotation and scaling can be expressed as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The four elements rs_{ij} are the multiplicative Rotation - Scaling terms in the transformation that involve only Rotation angles and scaling factors.

Elements trs_x and trs_y are the translational terms containing combinations of translation distances, pivot-point and fixed point coordinates, and rotation angles and scaling parameters.

The above matrix equation require 9 multiplication and 6 addition and the transformed coordinates are

$$x' = x \cdot rs_{xx} + y \cdot rs_{xy} + trs_x$$

$$y' = x \cdot rs_{yx} + y \cdot rs_{yy} + trs_y$$

Other Transformation

Additional Transformation used in the graphics packages are

i) Reflection
ii) Shear

Reflection

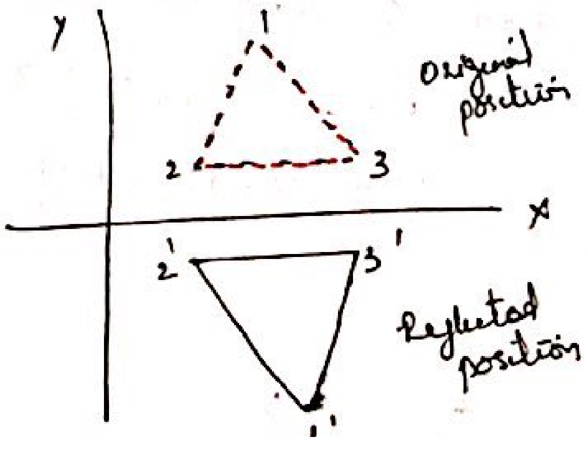
A reflection is a transformation that produces a mirror image of an object. The mirror image for a 2-D reflection is generated relative to an axis of reflection by rotating the object 180° about the reflection axis.

The axis of reflection can be choose in the xy plane or perpendicular to the xy plane.

when the reflection axis is a line in the xy plane, the rotation path about this axis is in a plane perpendicular to the xy plane. For the reflection axes that are perpendicular to the xy plane, the rotation path is in the xy plane.

Reflection about the line $y=0$, the x axis is accomplished with the transformation matrix

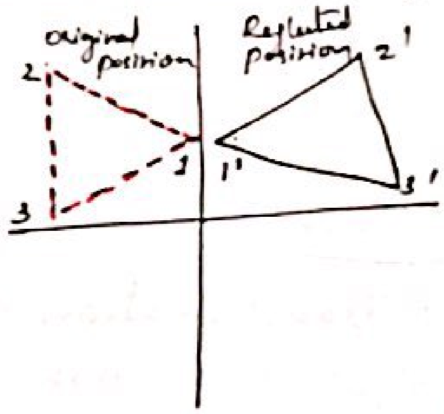
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



This transformation keeps x value the same but flips the y value coordinate position.

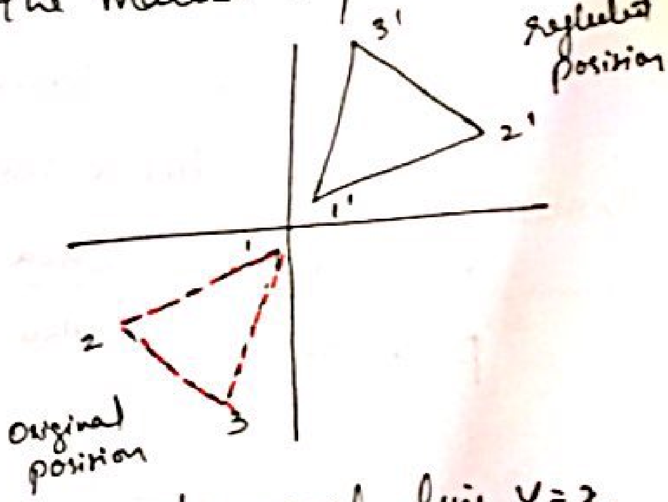
Reflection about y-axis flips x coordinate ~~which~~ while keeping y coordinates the same. The matrix for this transformation is

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



Reflection relative to an axis that is perpendicular to the xy plane & that passes through the coordinate origin flip both x and y. The matrix representation is

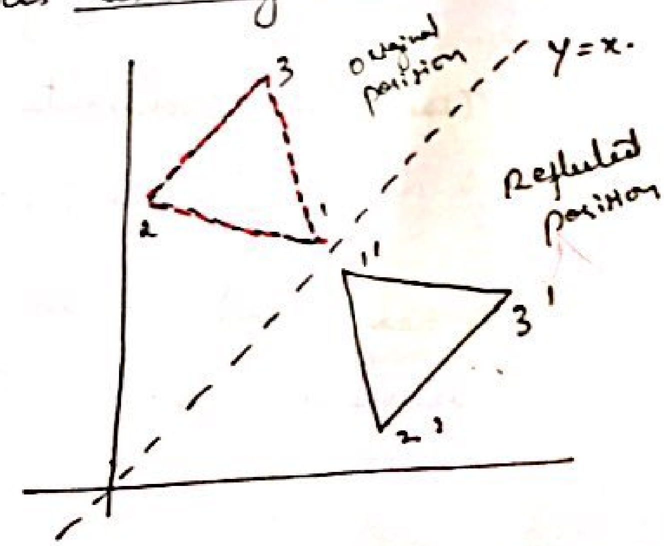
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Reflection about the axis as diagonal line $y = x$.

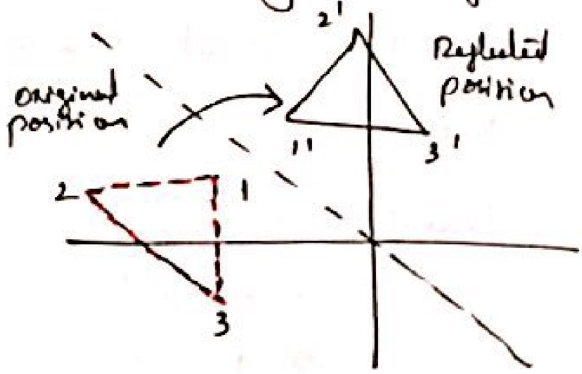
the reflection matrix is

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Reflection about the diagonal $y = -x$. The resulting transformation matrix is given by

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



Shear

A transformation that distorts the shape of an object such that the transformed shape appear as if the object were composed of internal layers that had been caused to slide over each other is called a shear.

Two common shearing transformations are those that shift x value coordinates & y-value coordinates.

An x-direction shear relative to the x-axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transform coordinate position as

$$x' = x + Sh_x \cdot y \quad , \quad y' = y .$$

A real number can be assigned to the shear parameter Sh_x.

Scanned with CamScanner

x- direction shear relation to other reference line $y = y_{ref}$
provide ~~with~~ transformation matrix as

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate position transformed as

$$x' = x + sh_x(y - y_{ref}), \quad y' = y.$$

A y-direction shear relative to the line $x = x_{ref}$ is generated with the transformation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

which generates transformed coordinate positions as

$$x' = x, \quad y' = sh_y(x - x_{ref}) + y$$

This transformation shift a coordinate position vertically by an amount proportional to its distance from the reference line $x = x_{ref}$.

Windowing concept.

A world coordinate area selected for display is called a window.

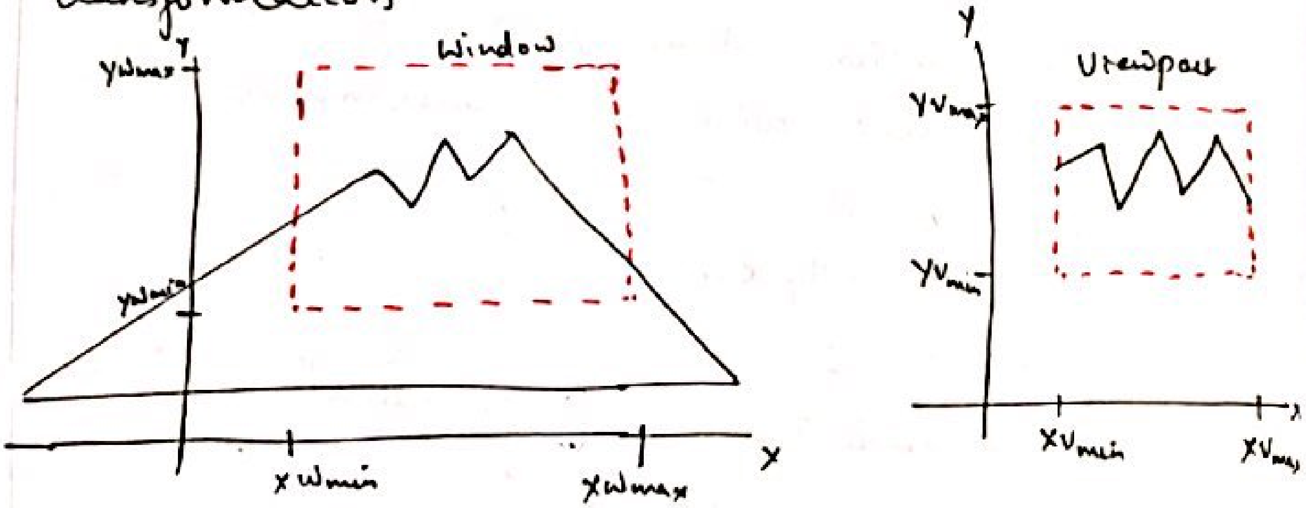
An area of the display device to which a window is mapped is called a viewport

The window defined what is to be viewed, and the

Viewport defines where it is to be displayed.

Windows and viewports are rectangles in standard position. with rectangle edges parallel to the coordinate axes

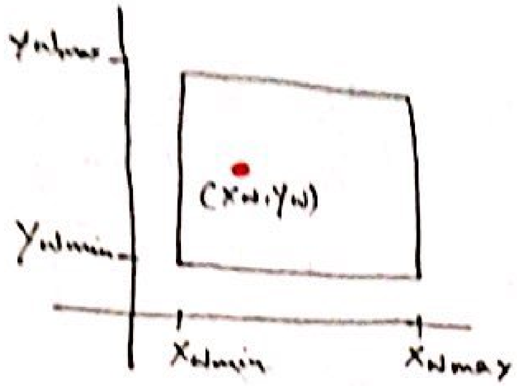
Mapping of a part of world coordinate scene to device coordinate is referred to as a viewing transformation or referred as window to viewport transformation

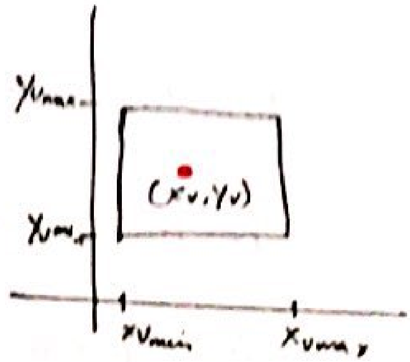


World coordinates Device coordinates

Window to Viewport Coordinate Transformation

In the window to viewport coordinate transformation object description is the window port is transformed to the normalized device coordinates. while transforming the object from window to viewport relative placement of object is maintained the same in the normalized viewing coordinates. If a coordinate position is at the centre of the viewing window, it will be displayed at the centre of the viewport.





A point at position (x_w, y_w) in window is mapped to viewport coordinates (x_v, y_v) so that relative position in the two areas are the same.

To maintain the same relative placement in the viewport as in the window, we require that

$$\frac{x_v - xv_{min}}{xv_{max} - xv_{min}} = \frac{x_w - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{y_v - yv_{min}}{yv_{max} - yv_{min}} = \frac{y_w - yw_{min}}{yw_{max} - yw_{min}}$$

Solving the above two equation for the viewport position (x_v, y_v) we have.

$$x_v - xv_{min} = xv_{max} - xv_{min}\left(\frac{x_w - xw_{min}}{xw_{max} - xw_{min}}\right)$$

$$x_v - xv_{min} = x_w - xw_{min}\left(\frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}\right)$$

$$\boxed{\therefore x_v = xv_{min} + x_w - xw_{min}(S_x) \quad \text{where } S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}}$$

Scanned with CamScanner

$$y_v - y_{vmin} = y_{vmax} - y_{vmin} \left(\frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} \right)$$

$$y_v - y_{vmin} = y_w - y_{wmin} \left(\frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}} \right)$$

$$y_v = y_{vmin} + y_w - y_{wmin} (S_y)$$

$$\boxed{\therefore y_v = y_{vmin} + (y_w - y_{wmin})\, S_y \quad \text{where } S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}}$$

Relative : proportion of the object are maintained if the scaling factors are the same ($S_x = S_y$). otherwise world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

Example :-
Find the viewport coordinate (x_v, y_v) with the window coordinates $(x_w, y_w) = (30, 80)$ and the min and max value of the window and viewport is given by.

$x_{wmin} = 20$
$x_{wmax} = 80$
$y_{wmin} = 40$
$y_{wmax} = 80$

$x_{vmin} = 30$
$x_{vmax} = 60$
$y_{vmin} = 40$
$y_{vmax} = 60$

equ is

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$\frac{x_v - 30}{60 - 30} = \frac{30 - 20}{80 - 20}$$

$$= \frac{x_v - 30}{30} = \frac{10}{60}$$

$$= x_v - 30 = 5$$

$$= x_v = 35$$

$$\frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}}$$

$$= \frac{y_v - 40}{60 - 40} = \frac{80 - 40}{80 - 40}$$

$$\frac{y_v - 40}{20} = \frac{40}{40}$$

$$y_v = 60$$

$$\therefore (x_v, y_v) = (35, 60)$$

TWO Dimensional clipping

Any procedures that identifies those portions of a picture that are either inside or outside of a specified region of space is referred as clipping. The region against which an object is clipped is called a clip window

Application of clipping

* Extracting part of a defined scene for viewing
* Identifying visible surfaces in 3-D view
* Creating object using solid modeling procedures.
* Displaying a multiwindow environment
* Drawing & painting operations that allow part of the picture to be selected for copying, moving, erasing or duplicating

In the viewing transformation, those picture parts that are within the window area are to be displayed. Everything outside the window is discarded. Clipping Algorithms are applied to the world coordinates, so that only contents of the window interior are mapped to device coordinates.

Types of Clipping :-

- point clipping
- Line clipping (straight-line segments)
- Area clipping (polygons)
- Curve clipping
- Text clipping

Line and polygon clipping routines are the commonly used clipping standards in the graphics packages.

Point clipping

A point $p = (x, y)$ is saved for display if the following inequalities are satisfied

$$xw_{min} \le x \le xw_{max}$$

$$yw_{min} \le y \le yw_{max}$$

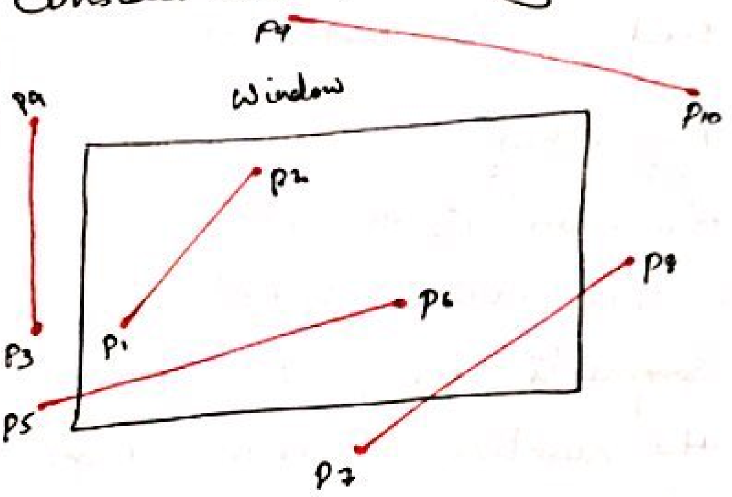
where the edges of the clip window (xw_{min}, xw_{max}, yw_{min}, yw_{max}) can be either the world coordinate window boundaries or viewport boundaries. If any one of the four inequalities is not satisfied, the point is clipped.

Line clipping

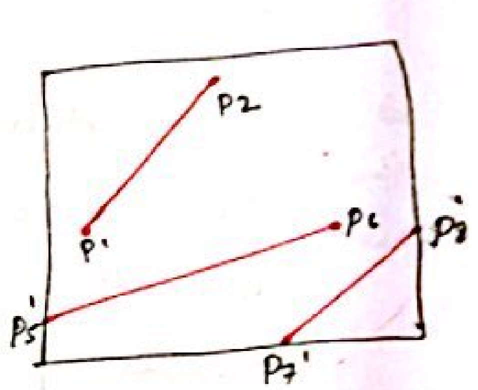
A line clipping procedures involves several parts. First, a line segment is tested to determine whether it lies completely inside the clipping window or it lies completely outside the clipping window.

If the first test fails then perform the intersection calculation with one or more clipping boundaries. The line is process through inside-outside tests by checking the line endpoints.

Consider the following lines



Before clipping



After clipping

In the above figure line P_1 to P_2 is saved because both endpoints are inside the clip window.

Line P_3 to P_4 and P_9 to P_{10} is discarded since both endpoints are outside the clip boundaries.

All other lines cross one or more clipping boundaries and may require calculation of multiple intersection points.

Scanned with CamScanner

All line segments fall into the following clipping categories

1. Visible :- Both end points of the line segment lie within the window

2. Non-visible :- when line lies outside the window. This will occur if the line segment from (x_1, y_1) to (x_2, y_2) satisfies any one of the following inequalities

$$x_1, x_2 > x_{max} \qquad y_1, y_2 > y_{max}$$

$$x_1, x_2 < x_{min} \qquad y_1, y_2 < y_{min}$$

3. Partially visible :- A line is partially visible when a part of its lies within the window

Cohen-Sutherland Algorithm

Cohen-Sutherland Algorithm is the most popular line clipping procedures. This method speeds up the processing of line segments by performing initial tests that reduce the number of intersections that must be calculated.

Every line end point in a picture is assigned a four digit binary code called a Region code that identifies the location of the point relative to the boundaries of the clipping rectangle.

Regions are set up in reference to the boundaries as shown in the following figure. Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window to the left, right top or bottom.

1001	1000	1010
0001	0000 window	0010
0101	0100	0110

Below 2nd Bit 1
Left 4th bit 1
ABRL
above first bit 1
Right bit is 3rd bit 1

The region above of window is 1000
The region below of window is 0100
The region left of window is 0001
The region right of window is 0010
Top left corner is 1001 (OR operation of above and left region of window)
Top right corner is 1010 (OR operation b/w right & above)
Bottom left corner is 0101 (OR operation b/w left & below)
Bottom right corner is 0110 (OR operation b/w below & right)

For any endpoint (x, y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

- First bit set 1 : point lies to left of window $x < x_{min}$
- Second bit set 1 : point lies to right of window $x > x_{max}$
- Third bit set 1 : point lies below (bottom) window $y < y_{min}$
- Fourth bit set 1 : point lies above (top) window $y > y_{max}$

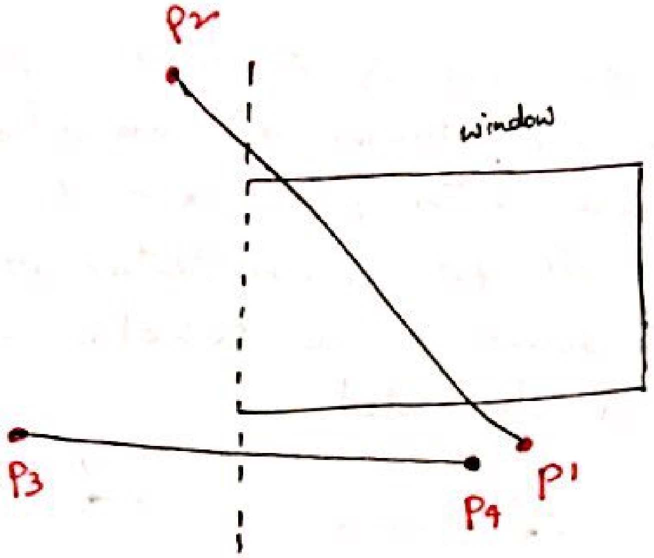
* Any lines that are completely contained within the window boundaries have a region code of 0000 for both endpoints and can trivially accept that line

* Any lines that have a 1 in the same bit position in the region code for each endpoints are completely outside the clipping window and trivally reject that line ie we can discard the line that has a region code of 1001 for one end point and a code of 0101 for the other endpoint. Both this endpoint are left to the clipping window

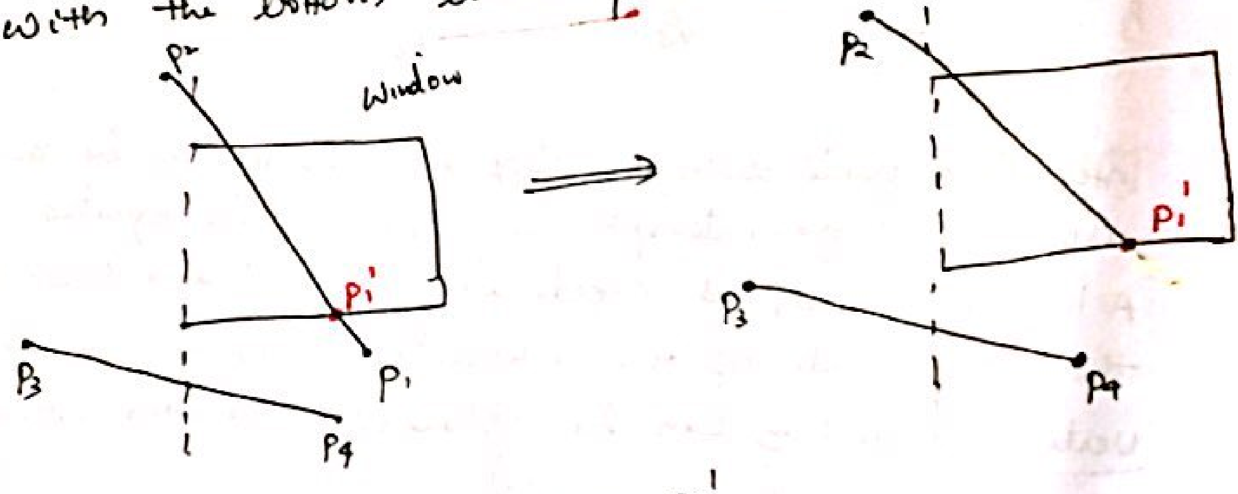
* The lines which is not completely inside or outside of the window is checked by performing logical AND operation with both region codes. If the result of AND operation is 0000 then part of the line may lie inside the window region. and the line segment cross the window edge

* Begin the clipping process for a line by comparing an outside endpoint to a clipping boundary to determine how much of the line can be discarded. Then the remaining part of the line is checked against the other boundaries and we continue until either the line is totally discarded or a section is found inside the window.

Example:-
Consider the line given in the following figure.
Starting with the bottom endpoint of the line from P_1 to P_2, P_1 is checked against the left, right and bottom boundaries and find that the point P_1 lies

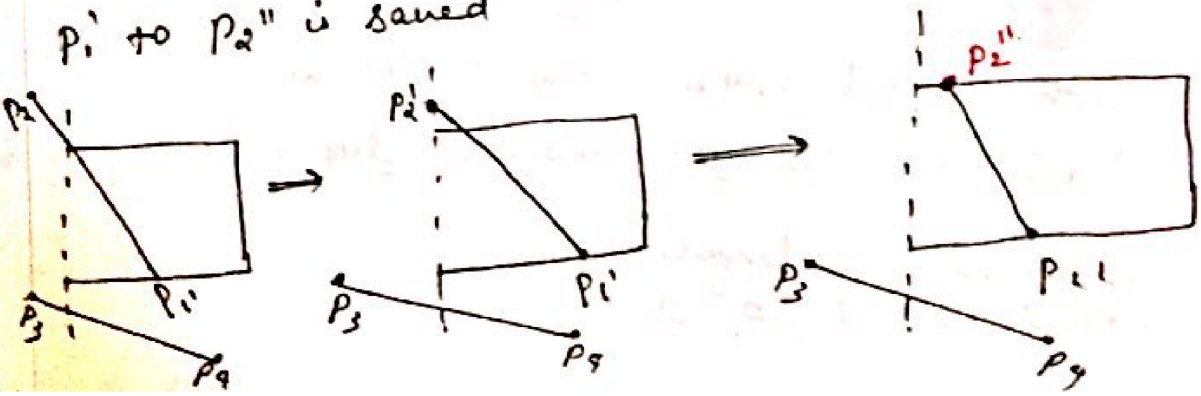


below of the clip window. Find the intersection point p_1' with the bottom boundary and discard the line p_1' to p_1

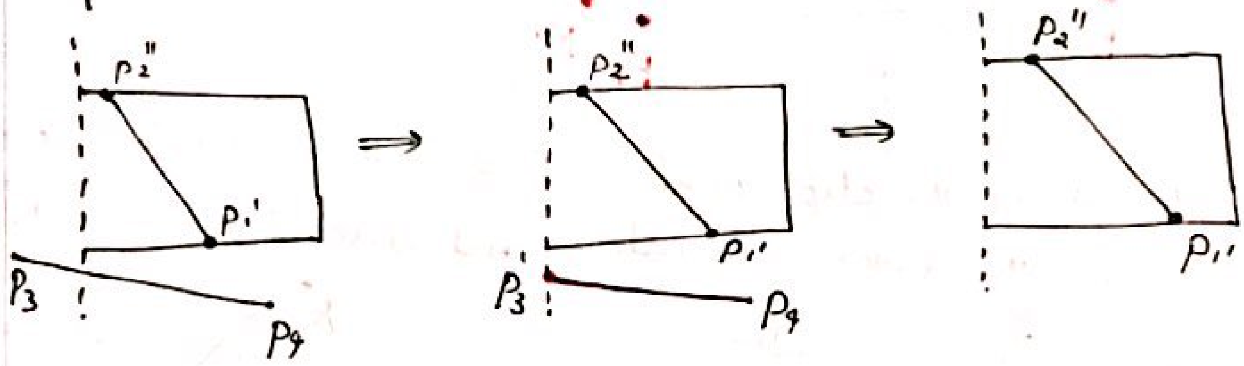


The line is reduced to P_2 to P_1'

Now the endpoint P_2 is outside the clip window, this point is checked against the boundaries and find that it is to the left of the window. Intersection point P_2' is calculated but this point is above the window. So the intersection calculation yield P_2'' and the line from P_1' to P_2'' is saved



For the second line the point P_3 is to the left of the clipping rectangle and determine the intersection P_3' and eliminate the line section from P_3 to P_3'. And by checking the region codes for the line section from P_3' to P_4 the line is found to be below the clip window and is discarded



Intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation.
A line with endpoint coordinates (x_1, y_1) and (x_2, y_2), the y coordinate of the intersection point with a vertical boundary can be obtained with the equation

$$y = y_1 + m(x - x_1)$$

where x value is set either to xw_{min} or xw_{max} and slope, m is calculated as $m = (y_2 - y_1)/(x_2 - x_1)$

The intersection with horizontal boundary, x coordinate can be calculated as

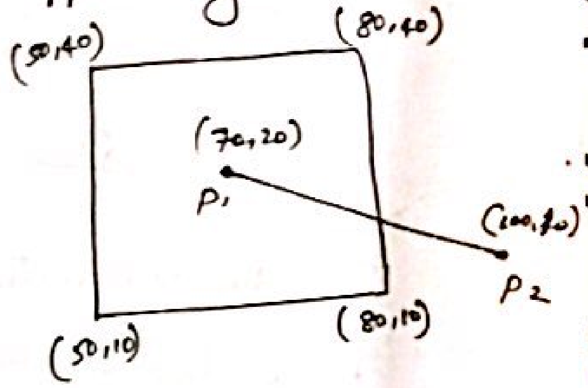
$$x = x_1 + \frac{y - y_1}{m}$$

with y set either yw_{min} or yw_{max}

Advantages of cohen sutherland line clipping

→ Simple
→ limited to rectangular regions
→ Extension of 3D clipping

Q Use the Cohen Sutherland algorithm to clip line $P_1(70,20)$ and $P_2(100,10)$ against a window lower left hand corner (50,10) and upper right hand corner (80,40)



Solution

$P_1 = (70,20)$

$P_2 = (100,10)$

left corner = (50,10)

right corner = (80,40)

Assign 4 bit binary outcode

point P_1 is inside the window so outcode of $P_1 = 0000$ and the outcode of $P_2 = 0010$ as P_2 is right of the window

AND operation of P_1 and P_2

$$\begin{array}{r} 0000 \\ 0010 \\ \hline 0000 \end{array}$$

The result of AND operation is zero. so line is partially visible.

slope of line P_1P_2 $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$

The intersection of line P_1P_2 with right edge of the window is point P_2.

Let the intersection point be (x,y).

$x = 80$; $y = ?$

$P_2(x_2,y_2) = P_2(100,10)$

Scanned with CamScanner

$$m = \frac{y - y_2}{x - x_2}$$

$$\frac{-1}{3} = \frac{y-10}{80-100} \Rightarrow \frac{-1}{3}x - 20 = y - 10$$

$$y - 10 = \frac{20}{3} \rightarrow y = \frac{20}{3} + 10 = 16.66$$

$\therefore$ the intersection point $P_3 = (80, 16.66)$

After clipping line P_1P_2 against window the new line is P_1P_3 with coordinates $P_1(70, 20)$ and P_3 $(80, 16.66)$

Midpoint Subdivision Algorithm

One of the disadvantage of Cohen-Sutherland is to find the intersection point of line with window boundary. Midpoint subdivision method is used to find the intersection point. The line segment is divided at its midpoint into two smaller line segments. The clipping categories the two new line segments into completly accepted or rejected or partially accepted. Each line segment which which needs to be partially accepted are divided again into smaller segments and then categories again.

The bisection (finding mid point) and categorization process continues until all line segments are completly accepted or rejected.

The midpoint coordinates (x_m, y_m) of a line segment joining $P_1(x_1, y_1)$ to $P_2(x_2, y_2)$ are given by

$$x_m = \frac{x_1 + x_2}{2} \quad \text{and} \quad y_m = \frac{y_1 + y_2}{2}$$

The Algorithm can be formalized as :-

For each endpoint :

a) If the end point lie in window, it is visible and the line accepted, process complete

b) If the end point lie outside the window, it is trivially invisible, and the line is rejected, process complete.

c) If the above two test fails, then Divide the line P_1P_2 at its midpoint P_m. Apply the previous tests to the two segments P_1P_m and P_mP_2. If P_mP_2 is trivally invisible then it is rejected. And continue with P_1P_m. ~~otherwise~~ this process continue until the intersection point of the line with the boundary is found.

Q A clipping window ABCD is specified as A(0,0) B(40,0) C(40,40) D(0,40). Using midpoint subdivision algorithm find the visible portion, if any, of the line segment joining the points P(-10,20) and Q(50,10)

Solution :-

The outcode of P is 0001 and Q is 0010. The both endpoint codes are not zero and their logical AND is zero, so that line cannot be Rejected

midpoint is

$$x_m = \frac{x_1+x_2}{2} = \frac{-10+50}{2} = 20$$

$$\therefore y_m = \frac{y_1+y_2}{2} = \frac{20+10}{2} = 15$$

outcode of midpoint $P_m(x_m, y_m)$ is 0000

Neither segment PP_m nor P_mQ is either totally visible or totally invisible. First consider the segment P_mQ and save the segment PP_m.

This subdivision process continues until we find an intersection points with window edge (40,y). The following Table show the subdivision work.

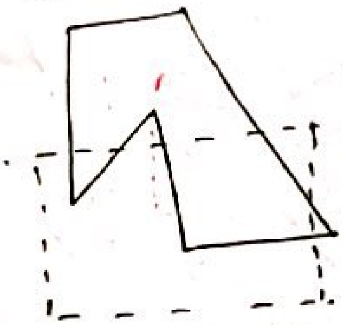
P	Q	P_m	Comment
(-10,20)	(50,10)	(20,15)	Save PP_m & Continue with P_mQ
(20,15)	(50,10)	(35,12)	Continue with P_mQ
(35,12)	(50,10)	(42,11)	continue with PP_m
(35,12)	(42,11)	(38,11)	Continue with P_mQ
(38,11)	(42,11)	(40,11)	This is the intersection point of line with Right Window edge
(-10,20)	(20,15)	(5,17)	Recall PP_m & Continue with PP_m
(-10,20)	(5,17)	(-3,18)	continue with P_mQ
(-3,18)	(5,17)	(1,17)	Continue with PP_m
(-3,18)	(1,17)	(-1,17)	Continue with P_mQ
(-1,17)	(1,17)	(0,17)	This is the intersection point of line with left window edge

Thus visible portion of line segment PQ is from (0,17) to (40,11)

14/4/18

Polygon clipping

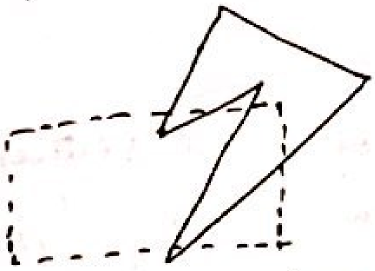
A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segment depending on the orientation of polygon to the clipping window. After polygon clipping a boundary area is been displayed.



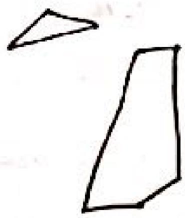
Before clipping

After line clipping of a polygon

polygon clipping algorithm generate one or more closed area. The output of the polygon. clipper should be a sequence of vertices that defines the clipped polygon boundary



Before clipping

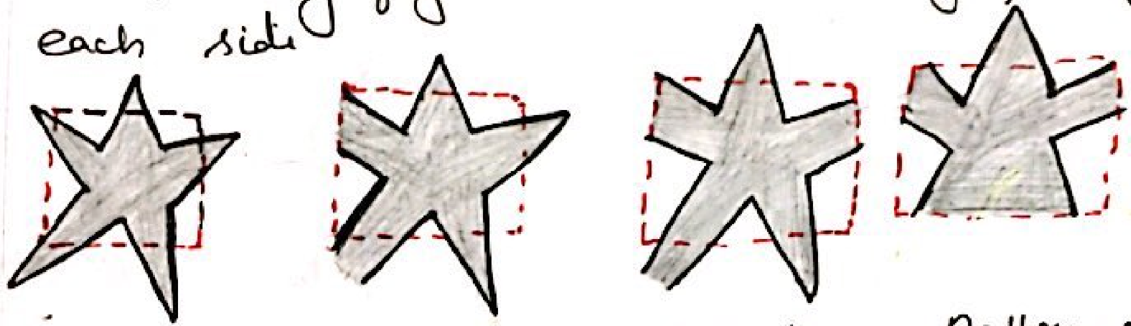


After clipping

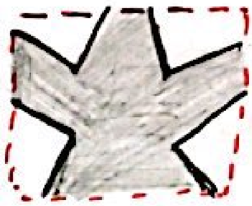
Sutherland Hodgeman polygon clipping

A polygon can be clipped correctly by processing the polygon boundary as a whole against each window edge. This is accomplished by processing all polygon vertices against each clip rectangle boundary in turn. First clip the polygon against the left rectangle boundary to produce a new sequence of vertices. The new set of vertices then passed to a right boundary clipper, a bottom boundary clipper and a top boundary clipper

The following figure shows the clipping of polygon in each side



Original polygon. | left clip | Right clip | Bottom clip.



Top clip.

At each step a new sequence of output vertices is generated and passed to the next window boundary clipper

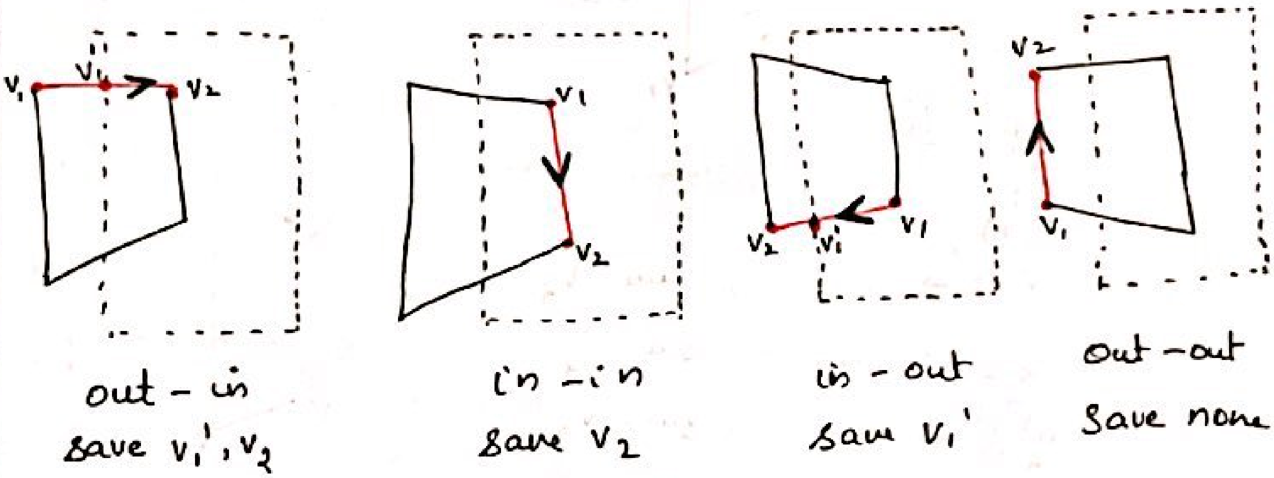
Following 4 tests are performed when each pair of adjacent polygon vertices is passed to a window boundary clipper

1) If the first vertex is outside the window boundary ~~only~~ and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.

2) If both input vertices are inside the window boundary only second vertex is added to the output vertex list.

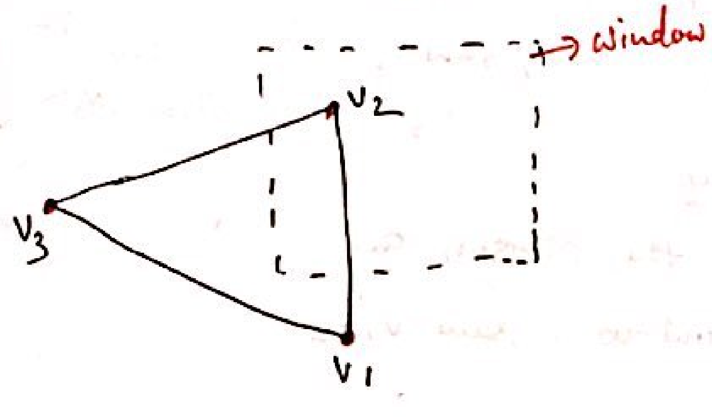
3) If the first vertex is inside the window boundary and the second vertex is outside only the edge intersection with the window boundary is added to the output vertex list

4) If both input vertices are outside the window boundary, nothing is added to the output vertex list



Implementing the algorithms requires a storage for an output list of vertices as a polygon.

Q Consider the following given polygon. clip the surface of the polygon which lies outside the clip window with respect to the vertices $V_1 V_2 V_3$



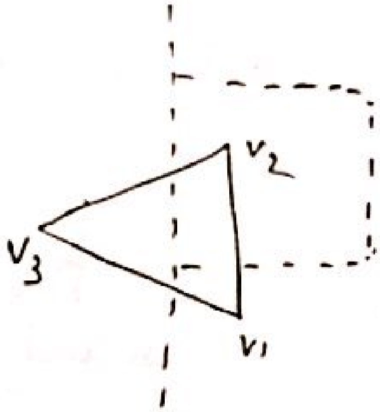
Left clip

The edges in the polygon are

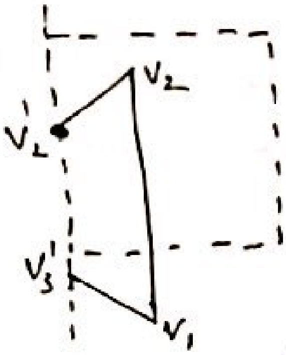
$V_1V_2 \rightarrow$ in-in, save V_2

$V_2V_3 \rightarrow$ in-out, save V_2'

$V_3V_1 \rightarrow$ out-in, save V_3' V_1



After left clip the polygon looks like



Right clip

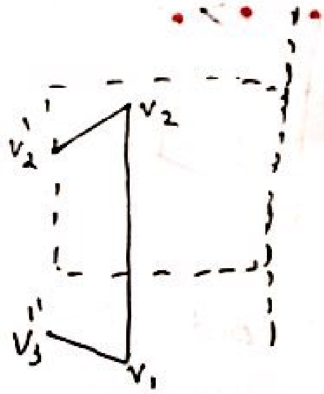
Edges in the polygon are

$V_1V_2 \rightarrow$ in-in, save V_2

$V_2V_2' \rightarrow$ in-in, save V_2'

$V_2'V_3' \rightarrow$ in-in, save V_3'

$V_3'V_1 \rightarrow$ in-in, save V_1



Top clip

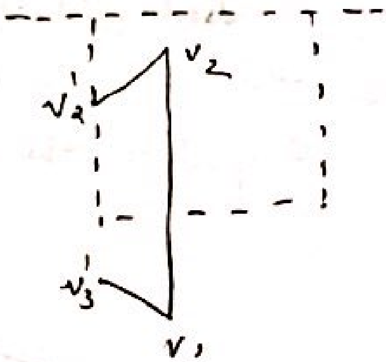
Edges of the polygon are

$V_1V_2 \rightarrow$ in-in, save V_2

$V_2V_2' \rightarrow$ in-in, save V_2'

$V_2'V_3' \rightarrow$ in-in, save V_3'

$V_3'V_1 \rightarrow$ in-in, save V_1



Bottom clip

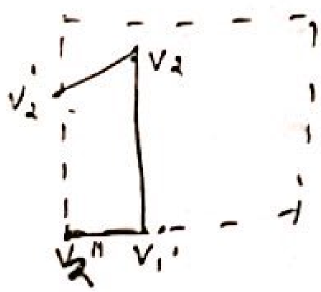
Edges of the polygon are

$V_1V_2 \rightarrow$ out-in, save $V_1'V_2$

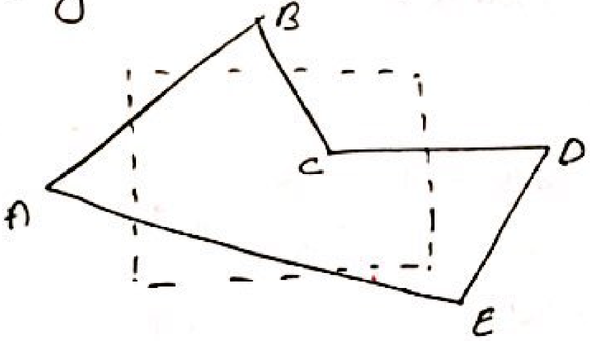
$V_2 V_2' \rightarrow$ in - in , save V_2'

$V_2' V_3' \rightarrow$ in - out , save V_2''

$V_3' V_1 \rightarrow$ out-out , save non



Q Consider a polygon ABCD, clip the polygon against the Rectangular Window



Left clip

Edges are

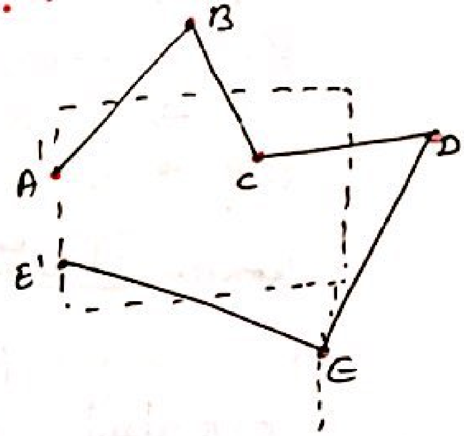
$AB \rightarrow$ out - in , save $A'B$

$BC \rightarrow$ in - in , save C

$CD \rightarrow$ in - in , save D

$DE \rightarrow$ in - in , save E

$EA \rightarrow$ in - out , save E'



clip Right

Edges are

$A'B \rightarrow$ in - in , save B

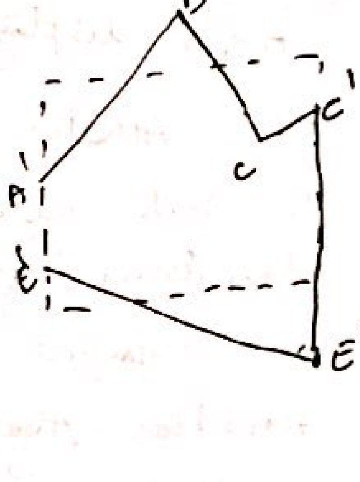
$BC \rightarrow$ in - in , save C

$CD \rightarrow$ in - out , save C'

$DE \rightarrow$ No vertex

$EE' \rightarrow$ out - in , save E''

$E'A' \rightarrow$ in - in , save A'



Bottom clip

$A'B \rightarrow$ in - in, save B

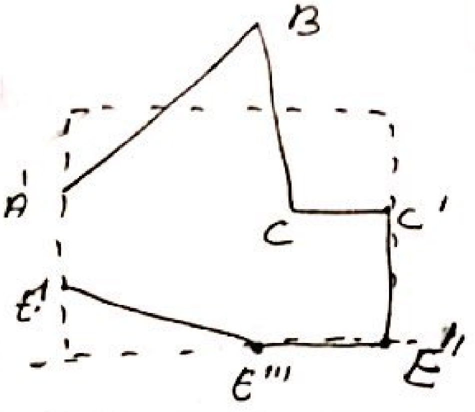
$BC \rightarrow$ in-in, save C

$CC' \rightarrow$ in-in, save C'

$C'E'' \rightarrow$ in-out, save C''

$E''E' \rightarrow$ out-in, save E''', E'

$E'A' \rightarrow$ in-in, save A'



Top clip

$A'B \rightarrow$ in-out, save B'

$BC \rightarrow$ out-in, save $B''C$

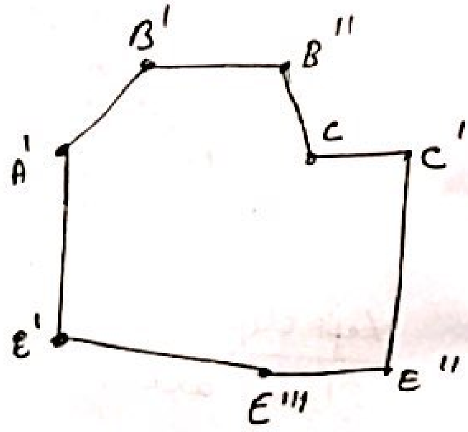
$CC' \rightarrow$ in-in, save C'

$C'E'' \rightarrow$ in-in, save E''

$E''E''' \rightarrow$ in-in, save E'''

$E'''E' \rightarrow$ in-in, save E'

$E'A' \rightarrow$ in-in, save A'



Advantage & Disadvantage of Sutherland-Hodgeman Algm

* All convex polygon are correctly clipped by the sutherland Algm
* But concave polygon clipping using sutherland Algm displayed with extraneous line.

Weiler Atherton polygon clipping

In weiler algorithm vertex processing procedures for window boundaries are modified so that concave polygons are displayed correctly. The basic idea in Weiler Atherton's Algorithm is that instead of always proceeding around the polygon edges as vertices are processed, sometimes want to follow the window boundaries. which path we follow depends on the polygon processing direction (clockwise or counterclockwise) and whether the pair of polygon vertices currently being processed represents an outside to inside pair or an inside to outside pair.

clipping window be initially called the clip polygon and the polygon to be clipped the subject polygon. Start with an arbitrary vertex of the subject polygon and trace around its border in the clockwise direction until an intersection ~~polygon~~ point of the polygon is reached.

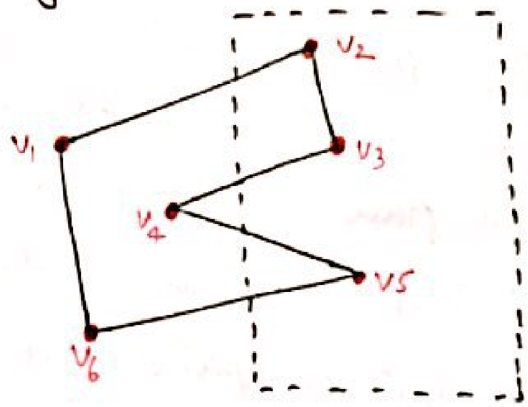
1. If the edge enters the clip polygon record the intersection point and continue to trace the subject polygon.

2. If the edge leaves the clip window, record the intersection point and make a right turn to follow the clip window in the same manner.

For clockwise processing of polygon vertices, we use the following rules.

- For an outside-inside pair of vertices, follow the polygon boundary.
- For an inside-outside pair of vertices, follow the window boundary in a clockwise direction

Q. Clip the following given polygon with the Weiler Atherton Algorithm.



1. Start with the starting vertex V_1, and move around the polygon. The first edge V_1V_2 is from outside - inside clip window. So the intersection point with the clip window is marked as V_1'

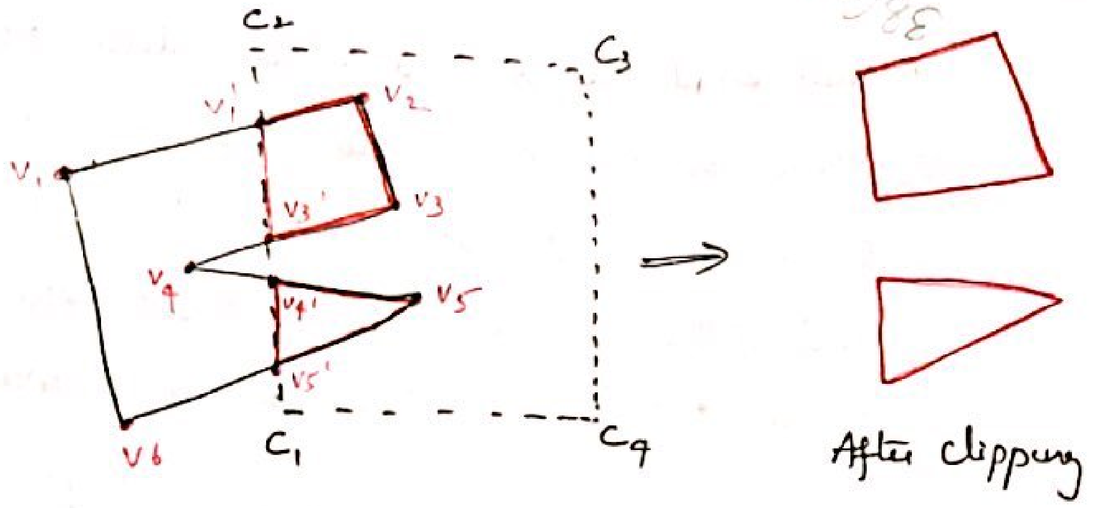
2. Moving around the edge V_2 to V_3. it is inside the clip window

3. Moving around the edge V_3 to V_4 it is inside to outside pair of vertices, then follow the window boundary ie find the intersection with the window boundary V_3' and make the right turn through the window boundary

4. Moving around the edge V_4 to V_5 it is outside-inside window so consider the intersection point V_4'.

5. Moving around edge V_5 to V_6 it is inside-outside so find the intersection point V_5' and moving around the window boundary.

Scanned with CamScanner



<u>Subject window</u>   <u>clip window</u>

Subject window	clip window
V_1	C_1
	V_5'
	V_4'
V_1'	V_3'
V_2	V_1'
V_3	C_2
V_3'	C_3
V_4	C_4
V_4'	
V_5	
V_5'	
V_6	

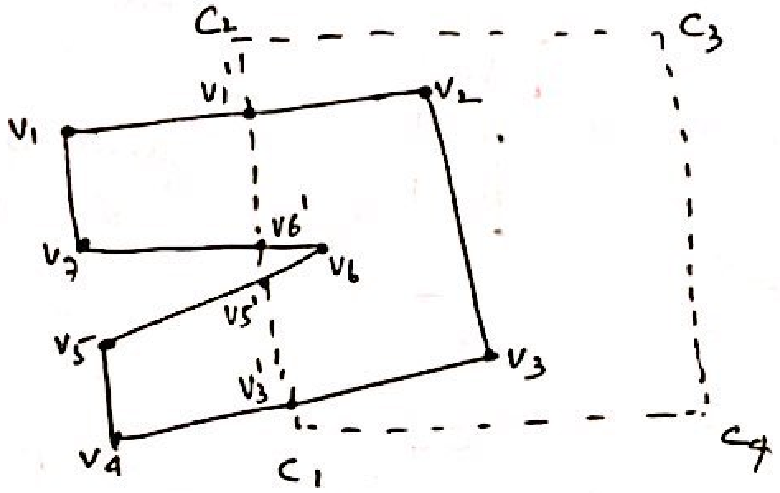
First boundary $= V_1' V_2 V_3 V_3' V_1'$

second boundary $=$ $V_4' V_5 V_5' V_4'$

Initially we are starting with first intersection in subject window and following the vertices till the next intersection is reached. If the next intersection is found in subject window match the second intersection point of subject window to the same intersection point present in clip window. Then follow the clip window till the first intersection point of the subject window is reached to get the closed polygon boundary.

• Then find the second intersection point V_4' and follow the subject window V_4' to the next intersection point V_5' and map V_5' to V_5' in the clip window and follow clip window to reach the initial intersection point till V_4' is reached and mark the boundary.

Q Consider the following polygon and clip the same using Weiler Atherton Algm



Consider edges.

$V_1V_2 \rightarrow$ out - in, save V_1'

$V_2 V_3 \rightarrow$ in - in, save V_2, V_3

$V_3 V_4 \rightarrow$ in-out, save V_3', V_5',

$V_4 V_5 \rightarrow$ out- out, None.

$V_5 V_6 \rightarrow$ out -in, save V_5', V_6

$V_6 V_7 \rightarrow$ in -out, save V_6', V_1'

$V_7 V_1 \rightarrow$ out -out, None

Three Dimensional object Representation

Graphics scene can contain many different kinds of object like trees, flowers, clouds, rocks etc. It is difficult to use a single method to describe all objects in graphics. because of the difference in characteristics of the object material.

Representation scheme for solid objects are divided into two broad categories

1) Boundary Representation
2) Space-partitioning Representation

Boundary Representation describe a 3-D objects as a set of surfaces that seperate the object interior from the environment eg: polygons.

Space-partitioning Representation are used to describe interior properties by partitioning the spatial region containing an object into a set of small, nonoverlapp contigous solids called cube. A common space partitioning representation is octree.

Polygon Surfaces

The most commonly used boundary representation for 3D graphics objects is a set of surface polygon that enclose the object interior. Many graphics system store all object descriptions as sets of surface polygons. This simplifies and speeds up the surface rendering and display of objects, since all surfaces are described with the linear equation. Due to this reason polygon descriptions are often referred to as standard graphics object

Polygon Table

A polygon surface is specify with a set of vertex coordinates and associated with the attribute parameters. As information for each polygon is input, data are placed into tables that are used in the subsequent processing, display and manipulation of the objects in the scene.

Polygon data table is organized into two groups.

1. Geometric data Tables
2. Attribute tables

Geometric data Table contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces

Attribute information of an object include parameters specifying the degree of transparency of the object and its surface reflexivity & texture characteristics.

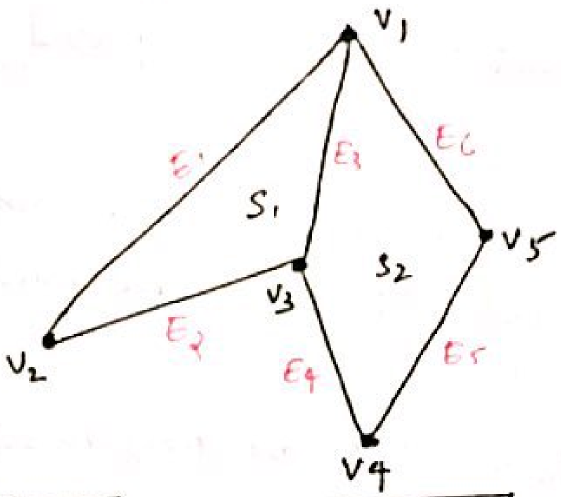
Geometric data Table consists of 3 parts

→ Vertex Table
→ Edge Table
→ polygon surface Table.

The edge table contains a pointer back to the vertex table to identify the vertices of each edge.

polygon Table contains pointers back into the edge Table to identify the edges for each polygon

Consider the following given polygon.



Vertex Table
$V_1 : x_1, y_1, z_1$
$V_2 : x_2, y_2, z_2$
$V_3 : x_3, y_3, z_3$
$V_4 : x_4, y_4, z_4$
$V_5 : x_5, y_5, z_5$
$V_6 : x_6, y_6, z_6$

Edge Table
$E_1 : V_1, V_2$
$E_2 : V_2, V_3$
$E_3 : V_3, V_1$
$E_4 : V_3, V_4$
$E_5 : V_4, V_5$
$E_6 : V_5, V_1$

polygon surface Table
$S_1 : E_1, E_2, E_3$
$S_2 : E_3, E_4, E_5, E_6$

Extra information can be added to the data tables for faster information extraction. We can expand the edge Table to include forward pointers into the polygon Table so that common edges between polygons could be identified more rapidly.

E_1	$: V_1, V_2, S_1$
E_2	$: V_2, V_3, S_1$
E_3	$: V_3, V_1, S_1, S_2$
E_4	$: V_3, V_4, S_2$
E_5	$: V_4, V_5, S_2$
E_6	$: V_5, V_1, S_2$

Additional geometric information is usually stored in the data table which includes

→ slope of each edge, m

→ Coordinate extends of each polygon ($x_{min}, x_{max}, y_{min}, y_{max}$)

Slopes can be calculated from the inputed vertices using $\frac{y_2 - y_1}{x_2 - x_1}$

By scanning the coordinate value the min & max value of x and y-coordinates can be ~~ea~~ identified for each polygon.

Some of the following Tests are performed by the graphics packages in the geometric data Tables are.

1) Check that every vertex is listed as an endpoint for at least two edges

2) check that every edge is a part of at least one polygon.

3) check that every polygon is closed

4) Check that each polygon has at least one shared edge

5) Check that if the edge tables contains pointers to polygon.

Plane Equation

To produce the display of 3D-object we must process the input data representation for the object through several procedures. These steps include transformation of the modeling and the world coordinate descriptions to viewing coordinates, then to device coordinates, identification of visible surface and the application of surface rendering (smoothening the polygon surface) procedures.

For some of these processes, we need information about spatial orientation of the individual surface components of the object. This information is obtained from the vertex coordinate values and the equations that describe the polygon planes.

The equation of the plane surface is expressed in the form

$$Ax + By + Cz + D = 0$$

where (x, y, z) is any point on the plane, and the coefficient A, B, C, D are the constants describing the spatial properties of the plane.

The values of the A, B, C, & D can be obtained by solving a set of 3 plane equations using the coordinate values for 3 noncolinear points in the plane (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3) and solve the equation for the ratio A/D, B/D and C/D. as

$$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1 \qquad k = 1, 2, 3$$

The solution for this set of equation can be obtained in determinant form using Cramer's Rule as

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

By expanding the determinants the plane coefficients is of the form

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2 z_3 - y_3 z_2) - x_2(y_3 z_1 - y_1 z_3) - x_3(y_1 z_2 - y_2 z_1)$$

plane equations are used to identify the position of spatial points relative to the plane surface of an object.

If $Ax + By + Cz + D \neq 0$ then point (x, y, z) is not on a plane

If $Ax + By + Cz + D < 0$ then point (x, y, z) is inside the surface

If $Ax + By + Cz + D > 0$ then point (x, y, z) is outside the surface

These inequality tests are valid in a Cartesian system, provided the plane parameters A, B, C, D were calculated using vertices selected in a counterclockwise order when viewing the surface in an outside-to-inside direction

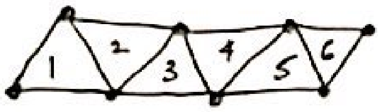
Polygon Meshes

Some Graphics packages provide several polygon functions for modeling objects. They are generally two polygon functions used.

* Triangle strip
* Quadrilateral mesh

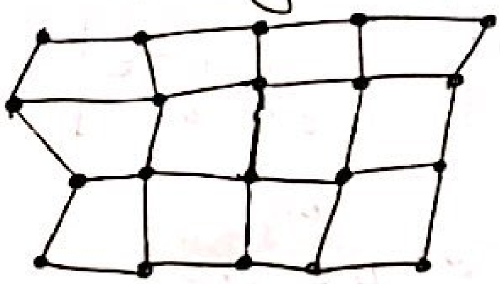
<u>Triangle strip</u> function produces (n-2) connected triangles

n = 8 then it produce 6 triangles



<u>Quadrilateral Mesh</u> generates a mesh of (n-1) x (m-1) quadrilaterals if the given coordinates for an n x m array of vertices.

Consider the following example.



5 vertices in column

4 vertices in Row

So the array is 4x5

4 quadrilateral in column and 3 quadrilateral in Row

4 x 5 → vertices

⇓

3 x 4 → quadrilateral

Basic 3D Transformation

Methods for geometric transformations and object modeling in 3-D are extended from 2-D method by including considerations for the z coordinate.

An object can be translate by specifying a 3-D translation vector, which determines how much the object is to be moved in each of the 3 coordinate direction

An object can be scaled with three coordinate scaling factors.

In rotation, rotation about an axis and the rotation about the plane is to be considered.

Translation

In a 3-D homogenous coordinate representation, a point is translated from position $p = (x, y, z)$ to position $p' = (x', y', z')$ with the following matrix operation

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

or

$$p' = T \cdot P$$

The Equation of Translation is given as

$$x' = x + t_x$$

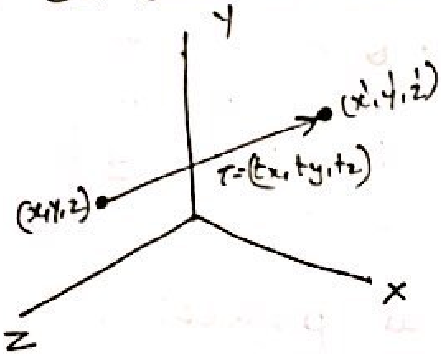
$$y' = y + t_y$$

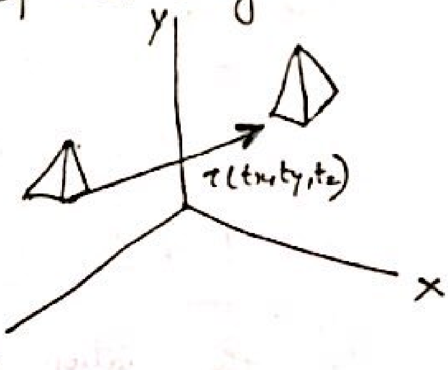
$$z' = z + t_z$$

The row matrix representation of 3-D translation transformation

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

* An object is translated in 3-D by transforming each of the defining points of the object. For an object which is represented as a set of polygon surfaces, we can translate each vertex of each surface.





* The inverse translation can be obtained by providing negative to the translation vectors like $(-t_x, -t_y, -t_z)$

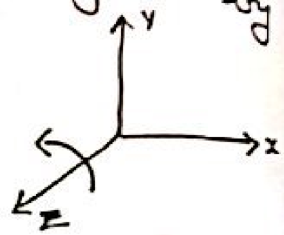
Rotation

To generate a 3-D rotation transformation for an object, we must decide an axis of rotation (axis by which the object is to be rotated) and the amount of angular rotation.

* positive angle rotation produce counterwise rotation about a coordinate axis
* Negative angle produce clockwise rotation about a coordinate axis.

Rotation about z-axis

The rotation of a object about z-axis is given by



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

parameter 'θ' specifies the rotation angle.

In homogenous coordinate system the 3-D z-axis rotation is expressed in <u>column matrix</u> form of <u>anticlockwise</u>

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

the equation is given as $p' = R(\theta) \cdot p$

The rotation 3D transformation in z-axis can be expressed in <u>row matrix</u> in <u>anticlockwise direction</u> is given as

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

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

The clockwise rotation $(-\theta)$ of the 3-D z axis in Column matrix form is expressed as

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

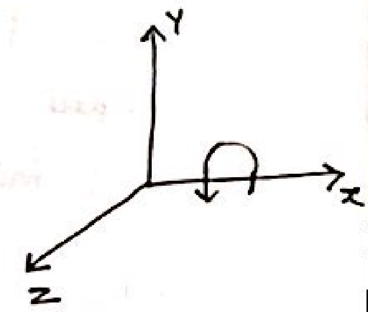
The clockwise rotation $(-\theta)$ of the 3-D z axis in row matrix form is expressed as

$$[x'\,y'\,z'\,1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about x- axis

Rotation about x-axis in 3-D is obtained from the Rotation about z-axis with transformation matrix of rotation about z-axis with a cyclic permutation of the coordinates parameter x, y and z ie $x \longrightarrow y \longrightarrow z \longrightarrow x$. Use in the z-axis rotation equation. and the equation becomes

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



where 'θ' specifies the rotation angle

In homogenous coordinate system the 3D- ~~object~~ x- axis rotation can be expressed in the column matrix form in anticlockwise direction ie (θ)

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

Rotation of 3D-object about x- axis in anticlockwise direction can be expressed in row matrix as follows:

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

X-axis 3D object Rotation in clockwise direction in column matrix can be expressed as

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

X- axis 3D object Rotation in clockwise direction in row matrix can be expressed as

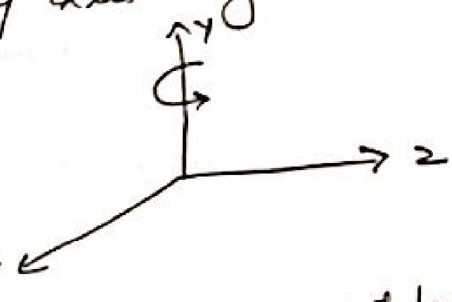
$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation of 3D-object about y. axis

Rotation of 3D-object about y axis in anticlockwise direction can be obtained the cyclic permutation in y-rotation of the coordinate parameter x, y and z

i.e $x \to y \to z \to x$

The equation of 3D rotation about y axis is given as



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

In homogenous coordinate systems the 3D-y axis rotation in anticlockwise direction expressed in column matrix is as follows

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

3D-y axis rotation in anticlockwise direction is expressed in row matrix is as follows.

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

3D-y axis rotation in clockwise direction $(-\theta)$ is expressed in column matrix as

Scanned with CamScanner

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

3D rotation in clockwise direction can be expressed in row matrix as

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

Scaling:

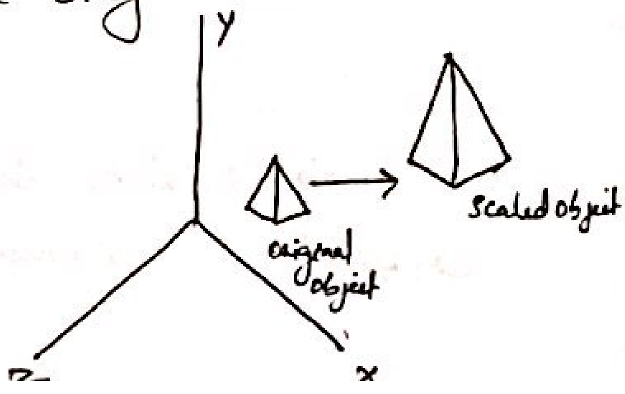
The matrix expression for the scaling transformation of a position $p = (x, y, z)$ relative to the coordinate origin can be written as

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

where the S_x, S_y and S_z are the scaling factors along x, y and z direction

The equation for the coordinate transformation for scaling relative to the origin are

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



① Scaling an object ~~with~~ changes the size of the object and repositions the object relative to the coordinate origin.

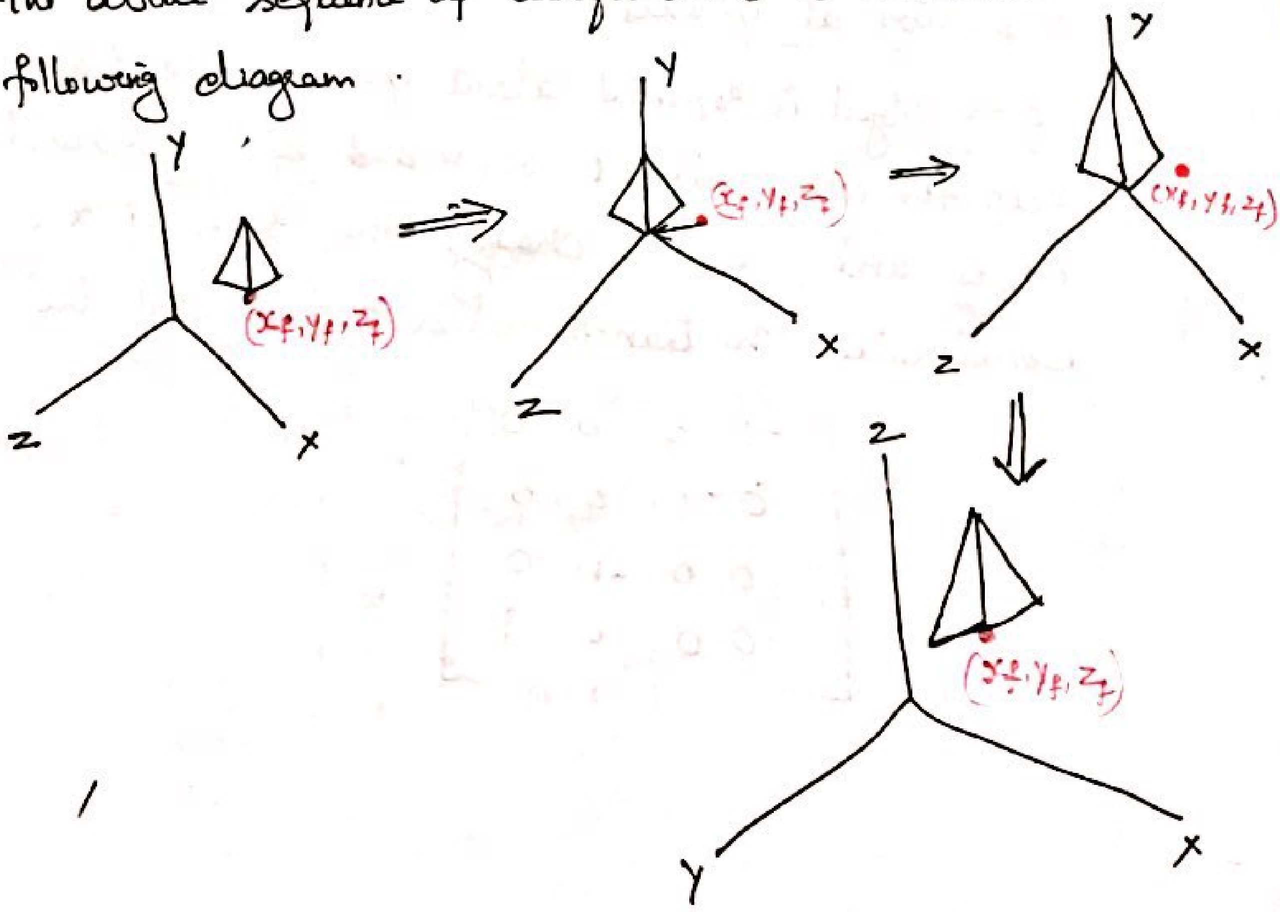
② If the scaling factors are not all equal, then the ~~dimension~~ dimension ~~of~~ in the object are change.

③ The original shape of an object can be preserve with a uniform scaling ($S_x = S_y = S_z$).

Scaling with respect to a selected fixed point (fixed position) (x_f, y_f, z_f) can be represented with the following transformation sequence:

1. Translate the fixed point to the origin
2. Scale the object relative to the coordinate origin
3. Translate the fixed point back to its original position.

The above sequence of transformation is demonstrated in the following diagram.



The matrix representation for an ~~arb~~ arbitrary fixed-point scaling can then be expressed as the concatenation of these translate - scale - translate transformation as

$$T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For perform inverse scaling, replace the scaling parameter s_x, s_y and s_z by their reciprocals as $1/s_x$, $1/s_y$ & $1/s_z$ in the scaling matrix.

Reflection

A 3-D reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

Reflection at y-axis :-

If the object is reflected about y-axis, we have to keep the magnitude of x, y and z coordinates as it is and need to change the sign of x & z coordinates. The transformation matrix will be

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

Reflection at x-axis :-

Value of x is not changed & y & z's sign get change. The matrix is

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

Reflection at z-axis :-

Value of z is not changed and x and y's sign is changed. And the transformation matrix is

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

Reflection through plane :-

Reflection through xy plane:-

In the reflection through xy plane only the z-coordinate value of the objects' position get change ie they are reversed in sign. The transformation matrix of reflection through the xy plane is

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

Reflection through yz plane :-

y & z is unchanged & sign of x is changed

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

Reflection through xz plane :-

Reflecting object about xz plane, x & z is unchanged and the sign of y is changed. The transformation matrix is given by

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

Shearing

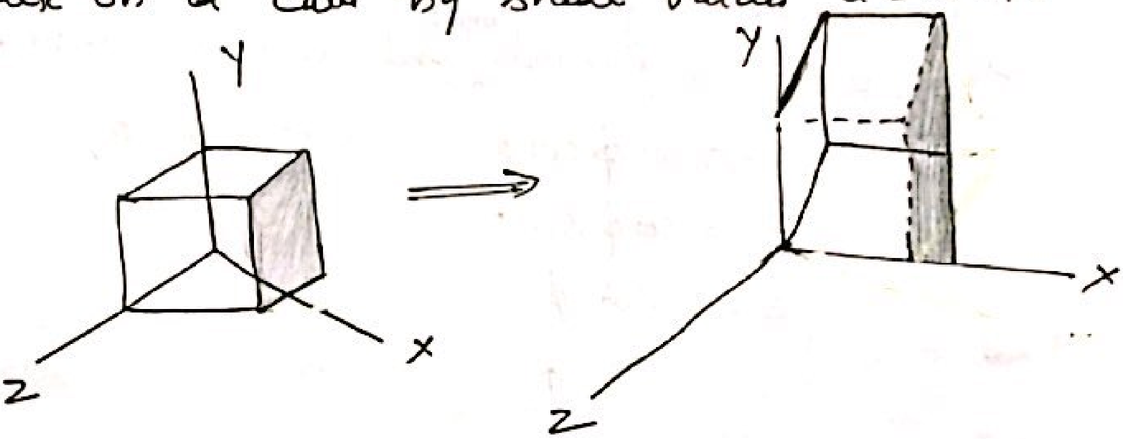
Shearing transformation can be used to modify object shapes. The following transformation produces a z-axis shear

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where a & b can be assigned to any real values.

The effect of the above transformation matrix is to alter x & y coordinate values by an amount that is proportional to the z value, while leaving z-coordinate as unchanged.

Following are the example of the effect of shearing matrix on a cube by shear values $a = b = 1$.



Quadric Surfaces.

A frequently used class of object are the quadric surfaces, which are described with second degree equations.

It includes.

→ sphere

→ Ellipsoids

→ Torus.

→ paraboloid

→ hyperboloid.

Quadric surfaces particularly sphere and ellipse are the common elements of graphics scene and are available in graphics packages.

Sphere

In Cartesian coordinates, a spherical surface with radius 'r' centered on the coordinate origin is defined as the set of points (x, y, z) that satisfy the following equation

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

The spherical surface can also be described in the parametric form using the latitude angle ϕ and longitude angle θ as

$$x = r\cos\phi\cos\theta \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r\cos\phi\sin\theta \qquad -\pi \le \theta \le \pi$$

$$z = r\sin\phi$$

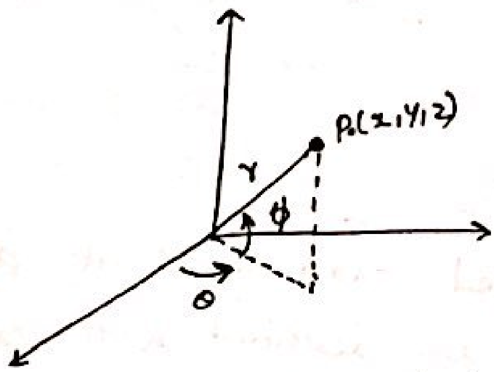


fig: parametric coordinate position (r, θ, ϕ) on the surface of a sphere with radius 'r'

Ellipsoid

An ellipsoidal surface can be described as an extension of a spherical surface, where the radius in three mutually perpendicular directions can have different values.

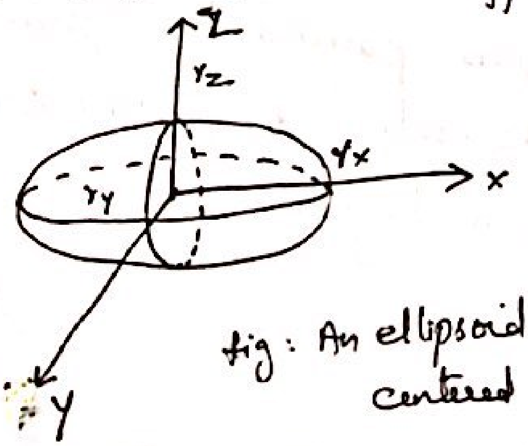


fig: An ellipsoid with radii r_x, r_y & r_z centered on the origin

The Cartesian representation for points over the surface of an ellipsoid centered on the origin is given by

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

Parametric representation for the ellipsoid in terms of the latitude angle ϕ and the longitude angle θ is given as

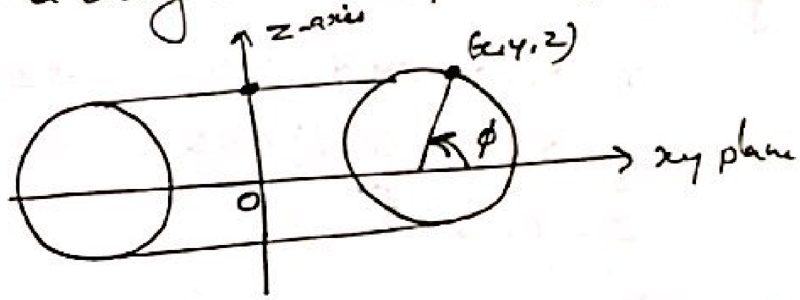
$$x = r_x \cos\phi \cos\theta \qquad -\pi/2 \le \phi \le \pi/2$$

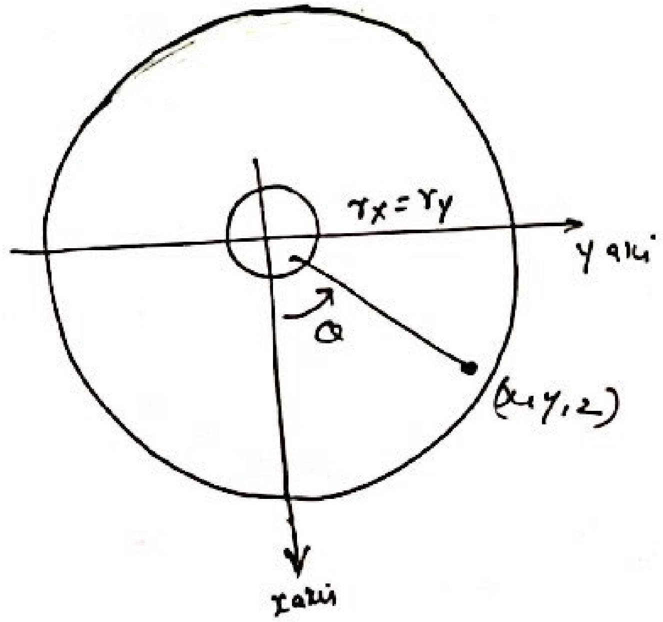
$$y = r_y \cos\phi \sin\theta \qquad -\pi \le \theta \le \pi$$

$$z = r_z \sin\phi$$

Torus

A Torus is a doughnut shaped object as given below





2/11/19

fig: A Torus with circular cross ~~region~~ section centered on coordinate origin

A Torus can be generated by rotating a circle or other conic shape about a specified axis. The cartesian representation for points over the surface of a Torus can be written in the form

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

where 'r' is any offset value.

parametric representation for a torus are similar to those of an ellipse, except that angle ϕ extends over $360°$. Using lattitude and longitude angles ϕ and θ, the torus surface can be describe as the set of points

$$x = r_x(r + \cos\phi)\cos\theta \quad , \quad -\pi \le \phi \le \pi$$

$$y = r_y(r + \cos\phi)\sin\theta \quad , \quad -\pi \le \theta \le \pi$$

$$z = r_z \sin\phi \, .$$

Projection

* Projection is the transformation of points in a coordinate system of dimension 'n' to a system of dimension less than 'n'.

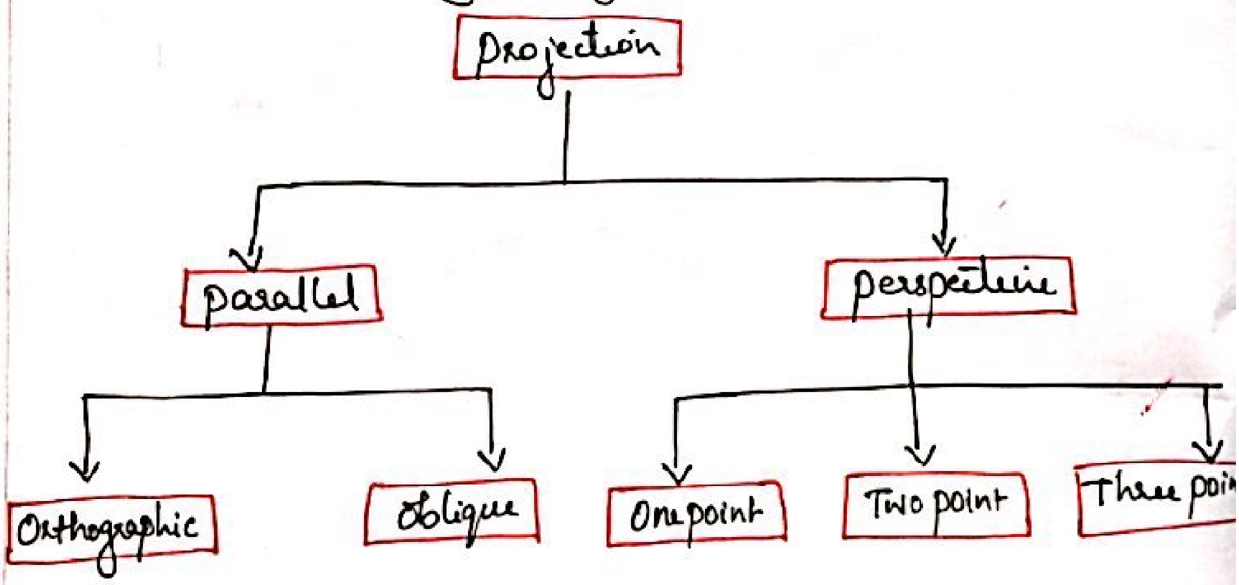
* Generally a 3-Dimensional object is projected to a 2-D object

* There are mainly 3 terms used in projection

i) Center of projection – point from where the projection is taken

ii) projection plane – plane on which projection of object is formed

iii) projector – lines emerging from the centre of projection and intersecting the projection plane after passing through a point on the object.



There are mainly two basic projection method.

* parallel projection
* perspective projection

• parallel projection :- • In this projection coordinate positions are transformed to the view plane along parallel lines ie projectors are ||el to each other

• Center of projection is at infinity

• Mainly used for scale drawing of 3D object

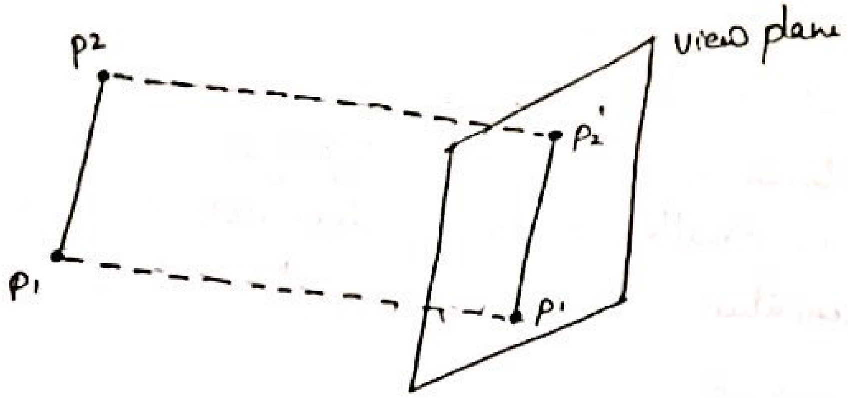


fig: parallel projection of an object to the view plane

• perspective projection :- • In this projection object position are transformed to the view plane along lines that converge to a point called the projection reference point (center of projection)

• The projected view is determined by calculating the intersection of the projection lines with the view plane

• Centre of projection is at finite distance

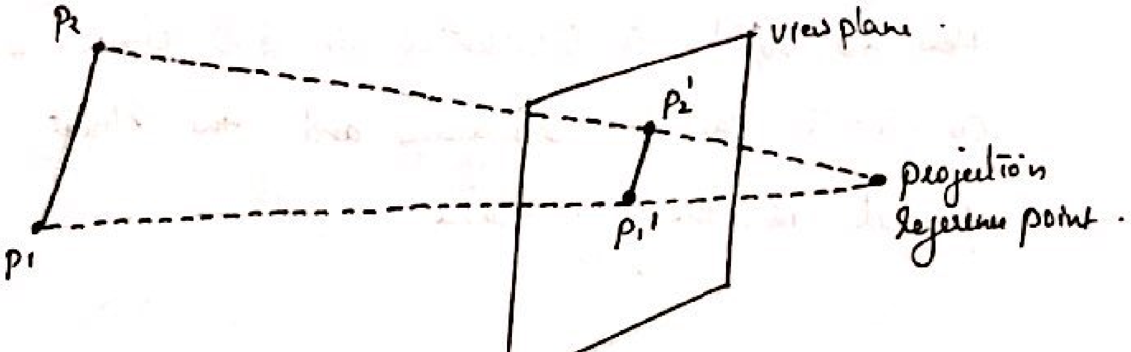
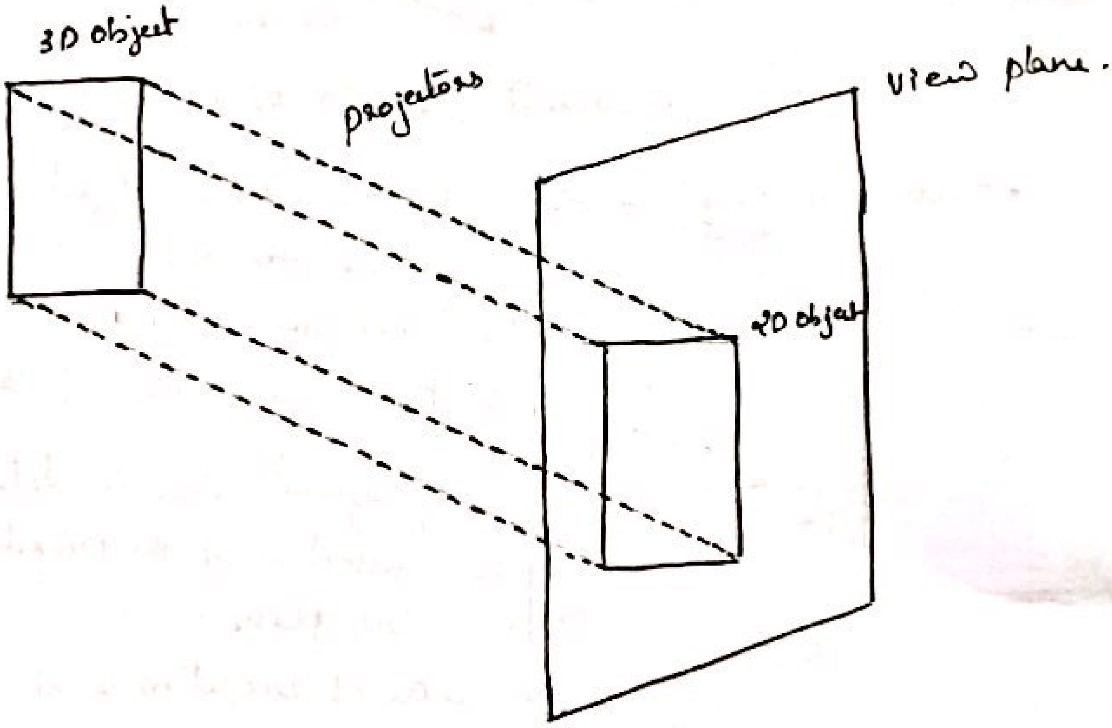


fig: prespective projection of an object to the view plane.

Scanned with CamScanner

Parallel Projections

- A parallel projection is formed by extending parallel lines from each vertex of the object until they intersect the plane of the screen.
- parallel projections preserves relative proportion of the object and this method is used to produce scale drawings of the 3-D objects.
- Accurate view of the various sides of the object can be obtained with the parallel projections.
- But the parallel projection does not give the realistic representation.



Here 3D object is represented in 2-D plane. so the z coordinate can be discarded and the object can be project in the 'xy plane.

Scanned with CamScanner

Let the direction of projection = (x_p, y_p, z_p)

Consider a point on the object (x_1, y_1, z_1)

Then the projected point to be determined as (x_2, y_2)

and the object is to determine in xy plane, so that

z-coordinate will be zero.

Then the equation for a line passing through the point

(x_1, y_1, z_1) and the direction of projection with

the use of parametric form as

$$x_2 = x_1 + x_p u$$
$$y_2 = y_1 + y_p u$$
$$z_2 = z_1 + z_p u$$

$z = 0$ then equation of z becomes

$$0 = z_1 + z_p u$$
$$z_p u = -z_1$$
$$\boxed{u = -\frac{z_1}{z_p}}$$

Sub u in 'x' & 'y' equation

$$x_2 = x_1 - z_1 \left(\frac{x_p}{z_p}\right)$$

$$y_2 = y_1 - z_1 \left(\frac{y_p}{z_p}\right)$$

The homogenous matrix form of the equation can be written as

$$\begin{bmatrix} x_2 & y_2 & z_2 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x_p}{z_p} & -\frac{y_p}{z_p} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or

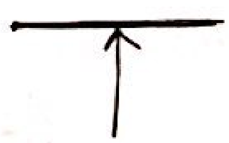
In coulum matrix it can be written as

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{-x_p}{z_p} & 0 \\ 0 & 1 & \frac{-y_p}{z_p} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

- The parallel projection can be specify with the projection vectors that defines the direction for the projection line ~~when the projection~~. They are of two types
  1) orthographic parallel projection
  2) oblique parallel projection

Orthographic parallel projection

- Projectors are perpendicular to the view plane in orthographic projection



- Since the projection is $\perp^{ar}$ it gives the true size and shape of a single ~~pla~~ plane face of the object.

* Orthographic projection do not change the length of the line segments which are parallel to projection plane.
* It is used to produce the front side, and top view of the object.
* Front side projection of an object is called elevation
* Top orthographic projection is called plan view.
* Matrix for projection onto the $x = 0$ plane is

$$P_x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here x column is all zero.

* Matrix for projection onto the $y = 0$ plane is

$$P_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

* Matrix for projection onto the $z = 0$ plane is

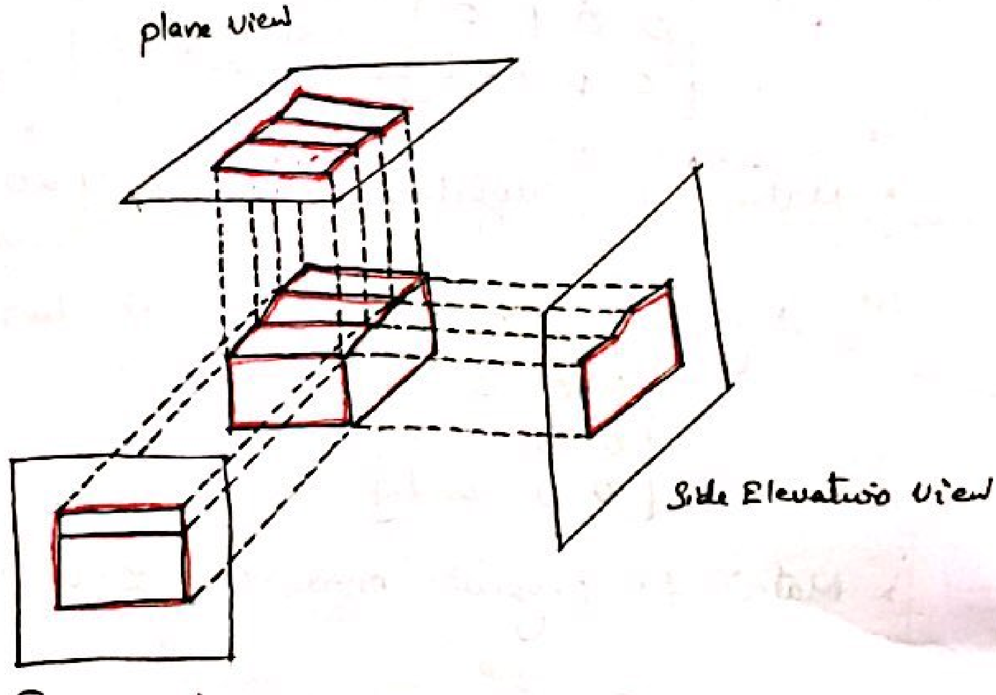
$$P_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Single orthographic projection does not give sufficient information to reconstruct the shape of the object. So multiple

orthographic projection are required called multiviews.
By combining multiple view like top, bottom, front, right and left side view of the object the whole object can be visually reconstructed.

Three views are used frequently.

* Front view - xy plane (when z=0)
* Right view - yz plane (when x=0)
* Top view - zx plane (when y=0)



* orthographic projections that display more than once face of an object is called axonometric orthographic projection

Subcategories of axonometric projections are

- Isometric
- Dimetric
- Trimetric

Isometric – Direction of projection makes equal angles with all three principal axis

Dimetric – Direction of projection makes equal angles with exactly two of the principal axis

Trimetric – Direction of projection makes unequal angle with the three principal axis

Oblique parallel projection

· In this projection the angle between the projector and the plane of projection is not equal to 90°.

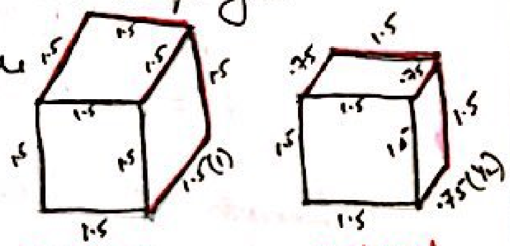
· projection are non-perpendicular to view plane.

· oblique parallel projection is seen in form of shadow of any object due to sunlight. Thus in this type of projection normally the shadow is displayed and body is not displayed.

Subcategories of oblique projection are

i) Cavalier projection

ii) Cabinet projection



Cavalier Cabinet

Cavalier projection is obtained when the angle between the oblique projectors and the plane of projection is 45°. Foreshortening factors of all 3 principal direction are equal ie $f = 1$

Cabinet projection, the angle between the oblique projector and the plane of projection is 63.43°. It is used to correct the distortion that is produced by cavalier projection

An oblique projection for which the foreshortening factor for edge $\perp^{ar}$ to the plane of projection is one-half. ie

$f = 1/2$.

Projection on xy plane with rays along a given direction

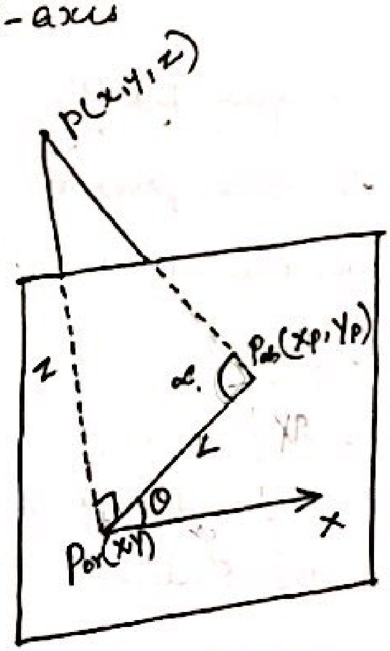
Consider a point $p(x, y, z)$ on the $z=0$ plane.

Let $P_{ob}(x_p, y_p)$ is an oblique projection of the point p

$P_{or}(x, y)$ is the orthographic projection of p on the $z=0$ plane.

θ - angle made by L with x-axis

Let L be the length of the line joining points $P_{ob}(x_p, y_p)$ and $P_{or}(x, y)$



The projection coordinates can be expressed in terms of x, y, L and θ as.

$$x_p = x + L\cos\theta$$
$$y_p = y + L\sin\theta$$

Length L depends on the angle α and the z coordinate of the point to be projected

$$\tan\alpha = \frac{z}{L} \Rightarrow L = \frac{z}{\tan\alpha} = zL_1 \quad \text{where } L_1 = \frac{1}{\tan\alpha}$$

when $z=1$, $L_1 = L$.

The oblique projection equations can be written as

$$x_p = x + z(L_1\cos\theta)$$
$$y_p = y + z(L_1\sin\theta)$$

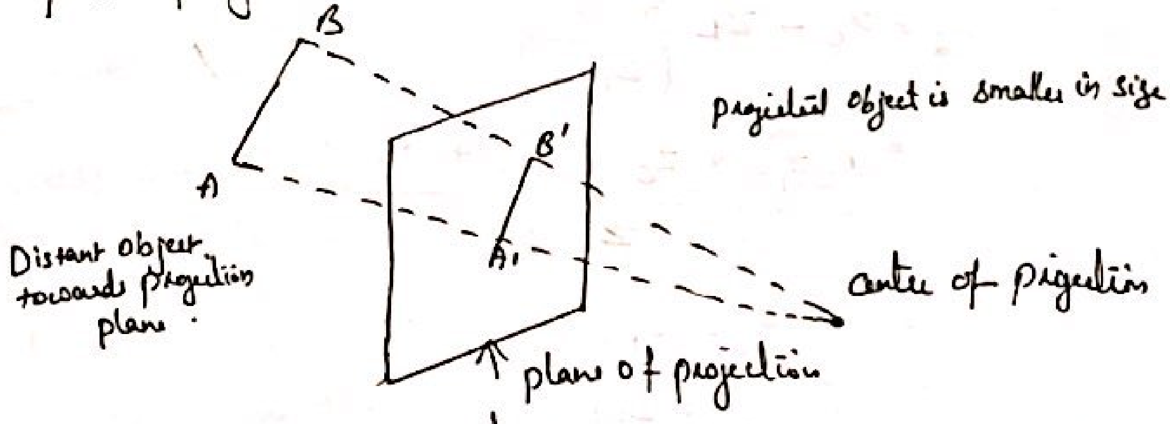
Considering $z_p = 0$ we get the following matrix

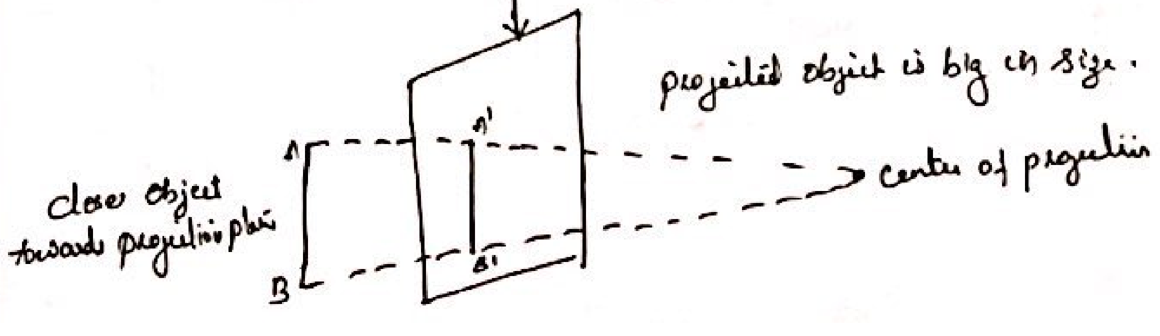
$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1\cos\theta & 0 \\ 0 & 1 & L_1\sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

This is the standard matrix for oblique projection onto $z=0$ plane.

Perspective projection.

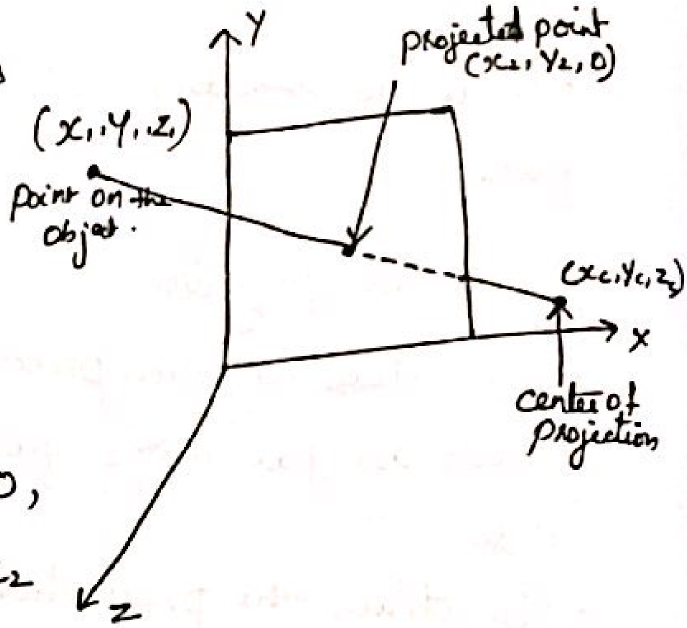
- Perspective projection preserves the property of an object which are far away from the viewer. It preserves the size.
- To obtain the perspective projection, the points along the ~~parall~~ projection lines which are not parallel to each other is transformed and converge to meet at a finite point known as projection reference point or centre of projection
- The projected view is obtained by calculating the intersection of the projection lines with the view plane.





Transformation Matrix for perspective projection

Consider the center of projection is at (x_c, y_c, z_c) and the point on the object is (x_1, y_1, z_1) then the parametric equation for the line containing these points can be given as



$$x_2 = x_c + (x_1 - x_c)u$$
$$y_2 = y_c + (y_1 - y_c)u$$
$$z_2 = z_c + (z_1 - z_c)u$$

where 'u' is a parameter

For projected point z_2 is 0, therefore the equation for z_2 can be written as

$$0 = z_c + (z_1 - z_c)u$$
$$u = \frac{-z_c}{(z_1 - z_c)}$$

Sub u in x_2 and y_2 equation & we got.

$$x_2 = x_c - z_c \frac{(x_1 - x_c)}{(z_1 - z_c)}$$
$$= \frac{x_c z_1 - x_c z_c - x_1 z_c + x_c z_c}{z_1 - z_c}$$
$$= \frac{x_c z_1 - x_1 z_c}{(z_1 - z_c)}$$

and $y_2 = y_c - \frac{z_c(y_1 - y_c)}{z_1 - z_c}$

$$= \frac{y_c z_1 - y_c z_c - z_c y_1 + z_c y_c}{z_1 - z_c}$$

$$= \frac{y_c z_1 - z_c y_1}{z_1 - z_c}$$

$$\therefore \boxed{\begin{aligned} x_2 &= \frac{x_c z_1 - x_1 z_c}{z_1 - z_c} \\ y_2 &= \frac{y_c z_1 - y_1 z_c}{z_1 - z_c}. \end{aligned}}$$

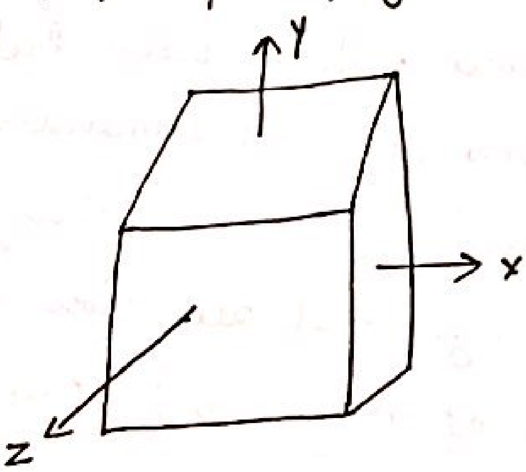
The homogenous matrix can be written as

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -z_c & 0 & x_c & 0 \\ 0 & -z_c & y_c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -z_c \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

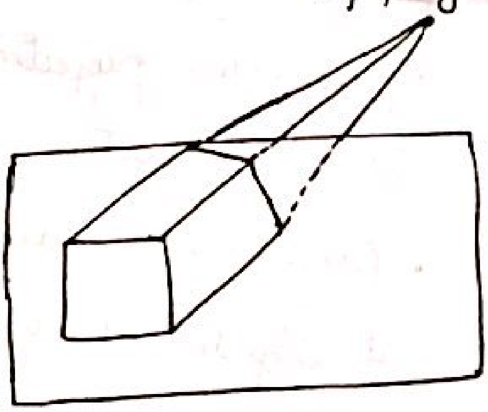
Vanishing points

- perspective projection produces realistic views but does not preserve relative proportion of object dimensions.
- projection of distant objects are smaller than the projection of objects of the same size that are close to the projection plane (centre of projection). This feature of perspective projection is known as <u>perspective foreshortening</u>

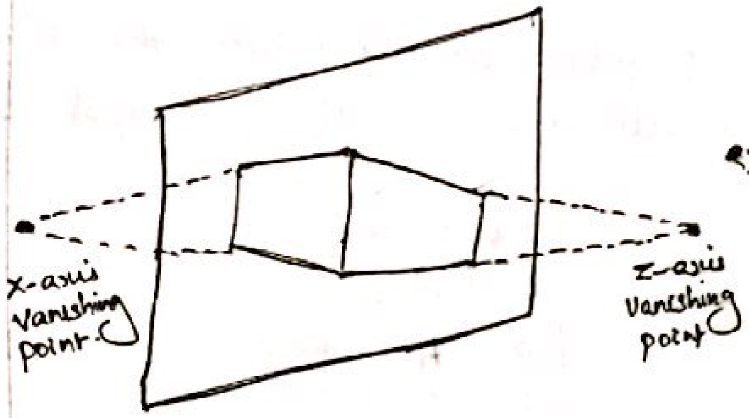
- Another feature of perspective projection is the illusion that, after projection certain set of parallel lines appear to meet at some point on the projection plane. These points are called <u>Vanishing point</u>.
- Each set of projected parallel lines have seperate Vanishing point.
- A scene can have any number of Vanishing points depending on how many sets of parallel lines are there in the scene.
- The vanishing point for any set of lines that are parallel to one of the principal axis of an object is referred to as a <u>principal Vanishing point</u> or <u>axis Vanishing point</u>
- The principal vanishing point with the orientation of the projection plane and perspective projections are classified as
  - → one point projection – only one principal axis intersect the plane of projection
  - → Two point projection – Two principal axis intersect the plane of projection
  - → Three point projection – Three principal axis intersect the plane of projection



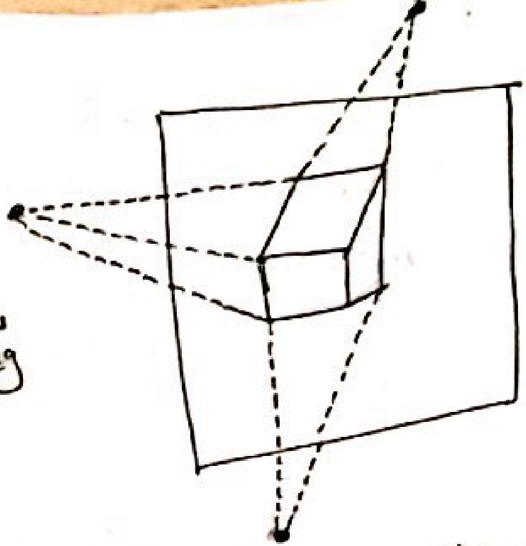
Coordinate description



One-point perspective projection



Two point perspective projection



Three-point perspective projection

<u>One-point perspective projection</u> occurs only one principal axis intersects the plane of projection. There are 3 types of one-point perspective transformation

i) when projectors are located at x-axis is given by

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1]\begin{bmatrix}1 & 0 & 0 & p\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Center of projection: $[-1/p \ 0 \ 0 \ 1]$

Vanishing point on x-axis $= [1/p \ 0 \ 0 \ 1]$

$$= [x \ \ y \ \ z \ \ (px+1)]$$

ii) when projectors are located on y-axis is given by

$$[x' \ y' \ z' \ 1] = [x \ \ y \ \ z \ \ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & q\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Center of projection $= [0 \ \frac{-1}{q} \ 0 \ 1]$

Vanishing point $= [0 \ \frac{1}{q} \ 0 \ 1]$

$$= [x \ \ y \ \ z \ (qy+1)]$$

iii) when projectors are placed at z-axis is given by

$$[x' \ y' \ z' \ 1] = [x \ \ y \ \ z \ \ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & r\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$= [x \ \ y \ \ z \ (rz+1)]$$

Center of projection $[0 \ 0 \ \frac{-1}{r} \ 1]$

Vanishing point $= [0 \ 0 \ \frac{1}{r} \ 1]$

Two point perspective projection occurs when the plane of projection intersects exactly two of the principal axis and is given by

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1]\begin{bmatrix}1 & 0 & 0 & p\\0 & 1 & 0 & q\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

$$= [x \quad y \quad z \quad (px+qy+1)]$$

Centre of projection on x-axis $= [-1/p \quad 0 \quad 0 \quad 1]$

Centre of projection on y-axis $= [0 \quad \frac{-1}{q} \quad 0 \quad 1]$

Vanishing point on x-axis $= [\frac{1}{p} \quad 0 \quad 0 \quad 1]$

Vanishing point on y-axis $= [0 \quad \frac{1}{q} \quad 0 \quad 1]$.

Three point perspective projection occurs when the projection plane intersects all three of the principal axis. ie none of the principal axis is parallel to the projection plane. The need of 3-point perspective transformation is to reconstruct the ~~sh~~ shape of a 3-D object. The matrix representation of 3-point perspective transformation is

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1]\begin{bmatrix}1 & 0 & 0 & p\\0 & 1 & 0 & q\\0 & 0 & 1 & r\\0 & 0 & 0 & 1\end{bmatrix}$$

$$= [x \quad y \quad z \quad (px+qy+rz+1)]$$

Centre of projection on x-axis $= [-1/p \;\; 0 \;\; 0 \;\; 1]$

Centre of projection on y-axis $= [0 \;\; -1/q \;\; 0 \;\; 1]$

Centre of projection on z-axis $= [0 \;\; 0 \;\; -1/r \;\; 1]$

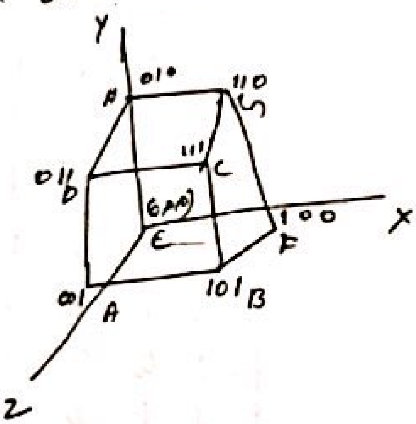
Vanishing point on x-axis $= [1/p \;\; 0 \;\; 0 \;\; 1]$

Vanishing point on y-axis $= [0 \;\; 1/q \;\; 0 \;\; 1]$

Vanishing point on z-axis $= [0 \;\; 0 \;\; 1/r \;\; 1]$

Examples on projection

Q To find orthographic projection of a unit cube onto the $x=0$, $y=0$ & $z=0$ plane.



Solution: The coordinate of the unit cube in matrix notation is as follows

$$\begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Transformation matrix for $x=0$ plane is $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Orthographic projection for $x=0$ is

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_x} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Orthographic projection onto $y=0$ plane is

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \underset{P_y}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic projection onto $z=0$ plane is

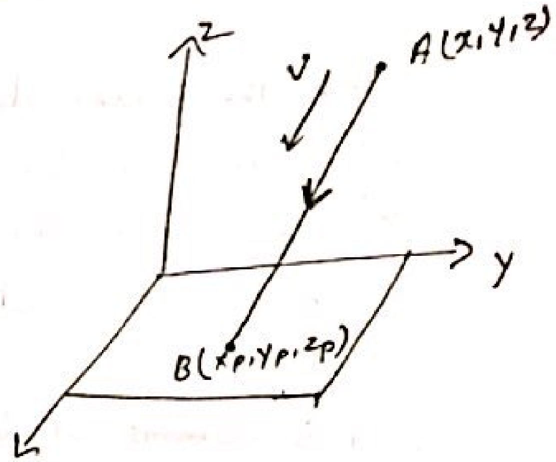
$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \underset{P_z}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Q. Derive the equation of parallel projection onto the xy plane in the direction of projection $V = a\hat{i} + b\hat{j} + c\hat{k}$

Solution :- Let $A(x, y, z)$ be any point and $B(x_p, y_p, z_p)$ is the parallel projection of A on xy plane. The vector $\vec{AB}$ is defined as

$$\vec{AB} = (x_p - x)\hat{i} + (y_p - y)\hat{j} + (z_p - z)\hat{k}$$

Since $\vec{AB}$ $\parallel^{el}$ to V, ie $\vec{AB} = tV$ where t is a constant



i.e. $(x_p - x)\hat{i} + (y_p - y)\hat{j} + (z_p - z)\hat{k}$
$= t(a\hat{i} + b\hat{j} + c\hat{k})$

i.e. $x_p - x = at$
$y_p - y = bt$
$z_p - z = ct$

Since the point $B(x_p, y_p, z_p)$ falls on xy plane $z_p = 0$

$\therefore 0 - z = ct$

$\rightarrow t = \frac{-z}{c}$

Sub t in above equatn

$\therefore x_p - x = a\left(\frac{-z}{c}\right)$

$x_p = x - \frac{az}{c}$

And $y_p - y = b\left(\frac{-z}{c}\right)$

$y_p = y - \frac{bz}{c}$

$z_p = 0$

The transformation matrix of 4×4 is of the form

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -a/c & 0 \\ 0 & 1 & -b/c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Q Find the transformation for

(a) Cavalier projection with $\theta = 45°$

(b) Cabinet projection with $\theta = 30°$

(c) Draw the projection for the unit cube for each transformation

Solution:-

a) for a ~~cabinet~~ cavalier projection there is no lines $\perp^{ar}$ to the xy plane. It make $\theta = 45°$ & $f = 1$ (Cavalier projection is equal in all 3 axis) $\left(\cos 45 = \frac{1}{\sqrt{2}} \quad \sin 45 = \frac{1}{\sqrt{2}}\right)$

The transformation required is

$$T = \begin{bmatrix} 1 & 0 & f\cos\theta & 0 \\ 0 & 1 & f\sin\theta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Downarrow \begin{array}{l} f = 1 \\ \theta = 45 \end{array}$$

$$T_1 = \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ cavalier projection transformation for $\theta = 45°$

(Column matrix of cavalier)

b) A cabinet projection is an oblique projection with $f = 1/2$ and $\theta = 30°$. we have. $\left(\cos 30 = \frac{\sqrt{3}}{2}\right)$ $\left(\sin 30 = \cancel{\#}\,\frac{1}{2}\right)$

$$T' = \begin{bmatrix} 1 & 0 & \frac{\sqrt{3}}{4} & 0 \\ 0 & 1 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ cabinet projection transformation for $\theta = 30°$.

(Column matrix of cabinet)

c) The vertices of the unit cube in homogenous coordinate system is given by.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$ (Row matrix for cube unit)

Apply Transformation matrix T_1 to coordinate matrix.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 1+\frac{1}{\sqrt{2}} & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 1 \\ 1+\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 1 \\ 1+\frac{1}{\sqrt{2}} & 1+\frac{1}{\sqrt{2}} & 0 & 1 \end{bmatrix}$$

(Row matrix of T_1)
Cavalier Transformation

The image coordinate of a cube are

$A : (0,0,0)$ $\quad E = \left(\frac{1}{\sqrt{2}}, 1+\frac{1}{\sqrt{2}}, 0\right)$

$B = (1,0,0)$ $\quad F = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$

$C = (1,1,0)$ $\quad G = \left(1+\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$

$D = (0,1,0)$ $\quad H = \left(1+\frac{1}{\sqrt{2}}, 1+\frac{1}{\sqrt{2}}, 0\right)$

To draw the cabinet projection, the image coordinate can be find by applying the transformation matrix to the coordinate matrix and the final matrix is given by.

Scanned with CamScanner

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{3}}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ \frac{\sqrt{3}}{4} & 1+\frac{1}{4} & 0 & 1 \\ \frac{\sqrt{3}}{4} & \frac{1}{4} & 0 & 1 \\ 1+\frac{\sqrt{3}}{4} & \frac{1}{4} & 0 & 1 \\ 1+\frac{\sqrt{3}}{4} & 1+\frac{1}{4} & 0 & 1 \end{bmatrix}$$

(Row matrix of Cabinet projection T')

The image coordinates are

$A' = (0,0,0)$ $\quad E' = \left(\frac{\sqrt{3}}{4}, 1+\frac{1}{4}, 0\right)$

$B' = (1,0,0)$ $\quad F' = \left(\frac{\sqrt{3}}{4}, \frac{1}{4}, 0\right)$

$C' = (1,1,0)$ $\quad G' = \left(1+\frac{\sqrt{3}}{4}, \frac{1}{4}, 0\right)$

$D' = (0,1,0)$ $\quad H' = \left(1+\frac{\sqrt{3}}{4}, 1+\frac{1}{4}, 0\right)$

Visible Surface detection Method.

In the graphics display systems we need to identify those parts of a scene that are visible from a chosen viewing Algorithm.

There are different approaches used to solve these problems and the different algorithms are created for the efficient identification of visible objects. for different types of application.

Some of the method have certain requirements as follows,

→ Some methods requires more space
→ Some require more computation power
→ Some methods are applied to special type of object.

Selection of each method depends on the following parameter

→ Depending on the complexity of the scene
→ Type of the object to be displayed
→ Available equipments.
→ Type of display to be generated (animated or static)

The Algorithms used to detect the visible surface are referred as visible surface detection method or hidden surface elimination methods.

VDM is classified as

Visible Surface Detection Method
- Object space Method
- Image space Method

Object Space Method	Image Space Method
• Visible surfaces are determined by compare objects and parts of the object to each other within the scene definition.	• Visibility is decided point by point at each pixel position on the projection plane.
• Method are developed in vector Graphics (Random Scan)	• Methods are developed in raster scan system.
• Accuracy for visible surfaces are more	• output is produced in less amount of time
• continuous operation	• Discrete in space.

The classification of Object space Method and Image space Method is as follows.

VSD

- Object space Method
  - Back Face Detection
- Image space Method
  - Depth Buffer Method (z-b
  - A-Buffer Algorithm
  - Depth Sorting Method
  - Scan line Method

Back Face Removal

'Back Face Removal method is a fast and simple method for identifying the back faces of a polyhedron is based on the 'inside-outside' tests.

A point (x, y, z) is "inside" a polygon surface with plane parameters A, B, C and D if

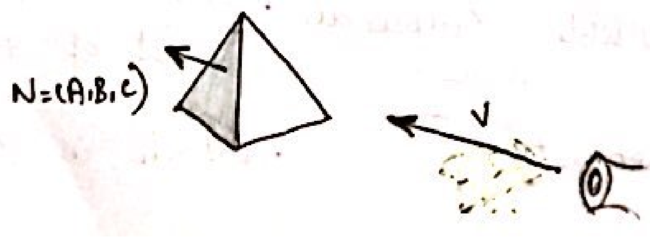
$$Ax + By + Cz + D < 0$$

when an inside point is along the line of sight to the surface, then the polygon surface is a back face.

The above test can be considered by the normal vector 'N' to the polygon surface.

The vector N has Cartesian components (A, B, C).

Consider a vector 'V' is a ~~vector~~ in the viewing direction from the camera (or eye) position as given in the following figure.



N → Normal vector i
polygon face

V → Vector in viewing
Direction

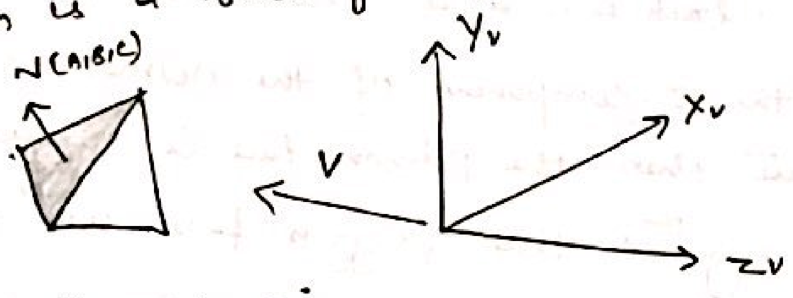
If $V \cdot N > 0$ then polygon face is backface
If $V \cdot N < 0$ then polygon face is frontface.

If the object description have been converted to projection coordinates and our viewing direction is parallel to the viewing z_v axis then the vector coordinates V can be written as $V = (0, 0, V_z)$ and

$$V \cdot N = V_z C$$

So we need to consider the sign of C & the z component of the normal vector N.

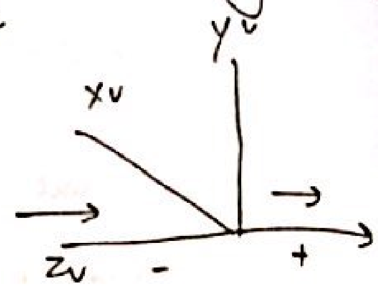
In a Right-handed viewing system with viewing direction is along the negative z_v axis, as shown in the figure, the polygon is a back face if $C < 0$.



Since V is in $-z_v$ axis,
the value $V_z C > 0$ (+ve) ie $-V_z x - C = V_z C$
$\therefore C < 0$ the polygon face is back face

NB:-
If the normal vector V has z component zero ie $C = 0$ then the surface cannot be seen by the viewer. In general if the normal vector z-component value $C \leq 0$ then the surface is backface in right handed system.

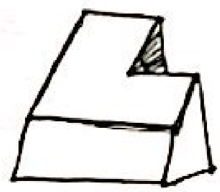
In case of the left handed systems the viewing direction is parallel to z-axis i.e along the positive Z-axis.



Back faces have normal vectors that points away from the viewing direction and is identified as $C \geq 0$

$\therefore V_z C = +V_z x + C = V_z C > 0$ i.e polygon has backface when $C \geq 0$ in lefthanded system.

Limitation of Back face Removal



- partial visible faces cannot be detected.

Note :-

If the Normal vector points towards the Viewer then the face is visible i.e a front face otherwise the face is hidden. (back face) and should be removed.

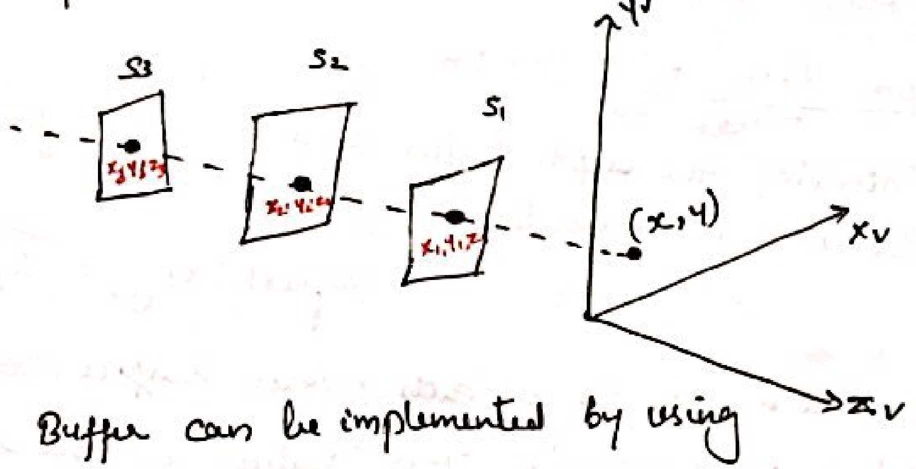
Detect the Z-component of the normal vector. If Z-component is positive then the polygon face is towards the viewer if it is negative then polygon face are away from the user.

Depth Buffer Method (Z-Buffer Method)

- Depth Buffer method is a commonly used image-space approach to detect the visible surface
- This method compares surface depth at each pixel position on the projection plane.
- This procedure is also called z-buffer method since the object depth is usually measured from the view

plane along the z-axis of a viewing system.

- This method usually applied to the scenes containing only polygon surfaces.
- In the projection transformation each polygon surface coordinate (x, y, z) is converted into projection point (x, y) on the view plane.
- For each pixel position (x, y) on the view plane, the object depth can be compared by comparing z-values.
- The following figure shows three surfaces at varying distance projection line from position (x, y) in a view plane taken as (x_v, y_v) plane.
- Surface S_1 is closest at this position, so its surface intensity value at (x, y) is saved.



- Depth Buffer can be implemented by using
  - i) Depth Buffer
  - ii) Refresh Buffer

<u>Depth Buffer</u> is used to store the depth value for each (x, y) position as the surfaces are processed.

<u>Refresh Buffer</u> stores the intensity value for position.

Procedure of Depth-Buffer method

- Initially all position in the depth buffer are set to 0 (minimum depth) and the refresh buffer is initialized to background intensity.
- Each surface in the polygon is processed, one scan line at a time and calculate the depth value (z-value) at each pixel (x,y) position.
- The calculated depth is compared to the previously stored value in the depth buffer at that position.
- If the calculated depth is greater than the previous value stored in the depth buffer, then the new depth value is stored in the buffer and the surface intensity in that position is determined and place the intensity value in the refresh buffer for the same pixel position (x,y)

Depth-Buffer Algorithm :-

1. Initialize the depth buffer and refresh buffer so that for all buffer position (x, y)

$$depth(x,y) = 0 \quad , \quad refresh(x,y) = I_{background} .$$

2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility

   - calculate the depth z for each (x,y) position on the polygon
   - If $z > depth(x,y)$ then set

$$depth(x,y) = z, \quad refresh(x,y) = I_{surf}(x,y)$$

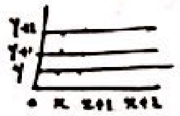
where $I_{backgnd}$ is the value of background intensity and $I_{surf}(x,y)$ is the projected intensity value for the surface at pixel position (x,y).

After processing all surfaces, the depth buffer contains depth values for the visible surfaces and the refresh buffer contains the corresponding intensity value for those surfaces.

Depth value (Z-value) for a surface position (x,y) are calculated from the plane equation for each surface

$$z = \frac{-Ax - By - D}{C}$$

- x value and y-value of the adjacent scan line is differ by 1



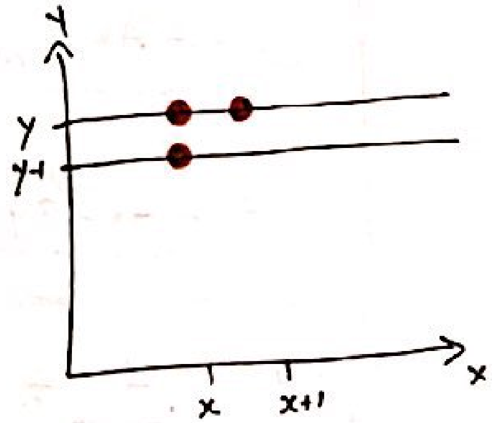
- If the depth of the position (x,y) has been determined to be z, then the depth z' of the next position $(x+1, y)$ along the scanline can be obtained as

$$z' = \frac{-A(x+1) - By - D}{C}$$

$$= \frac{-Ax - A - By - D}{C}$$

$$= \frac{-Ax - By - D}{C} - \frac{A}{C}$$

$$= z - \frac{A}{C}$$



$$\therefore \boxed{z' = z - \frac{A}{C}} \text{ at } (x+1, y)$$

Scanned with CamScanner

The Ratio $-A/C$ is constant for each surface.

- Succeeding depth value across the scan line are obtained from preceding value with a single addition with A/C.
- On each scan line :- the depth value is calculated on the left edge of the polygon that intersects the scan line.

The depth value (z-value) at the position $(x, y-1)$ is calculated with the slope.

$$\text{slop } m = \frac{y-y'}{x-x'} \quad \Big| \; y' = y-1$$

$$= \frac{y-y+1}{x-x'} = \frac{1}{x-x'}$$

$$\text{ie } x-x' = \frac{1}{m}$$

$$x' = x - \frac{1}{m}$$

$$z'_{y-1} = \frac{-A\left(x-\frac{1}{m}\right) - B(y-1) - D}{C}$$

$$= \frac{-Ax + \frac{A}{m} - By + B - D}{C}$$

$$= \frac{-Ax - By - D}{C} \quad \frac{+\frac{A}{m} + B}{C}$$

$$\boxed{z'_{y-1} = z + \frac{\frac{A}{m} + B}{C}}$$

When $m = \infty$ (infinity) then $z'_{y-1} = z + \frac{B}{C}$

Note :-

If viewer is viewing through +ve z-axis then the surface close to the viewer is having larger z-value ie surface close to the viewer has z_{max} value.

And if viewer is viewing through -ve z-axis then the surface close to the viewer is having smaller z-value ie surface ie surface close to the viewer has z_{min} value.

• Alternative approach used in the Z-Buffer Algorithm is Bresenham's method of Algorithm.

Disadvantage

• Time consuming
• Requires two additional buffer and hence need a large memory.
• Deals only with the opaque surfaces not more than one surface

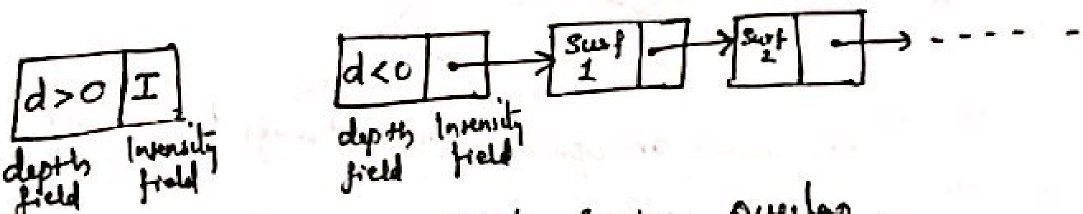
A-Buffer Method

A-Buffer Method is the extension of z-buffer method.

• A Buffer method expands the depth buffer so that each position in the buffer can reference a linked list of surfaces.

• More than one surface intensity can be taken into consideration at each pixel positioned and object edges are antialiased (smoothening)

• Each position in the A-buffer has two field
→ depth field - stores a positive or negative real number
→ Intensity field - stores surface intensity information or a pointer value.



Single-surface overlap

Multiple surface overlap.

• If depth field is d>0, the number stored at that position is the depth of a single surface overlapping. The intensity field store RGB component of the surface color at that point & percentage of pixel coverage

• If the depth field value is negative ie $d<0$, then the number stored at that position indicates the multiple surface contribution to the pixel intensity.

• Data for each surface in the linked list include

→ RGB intensity component
→ percentage of transparency
→ depth
→ percentage of area coverage
→ Surface identifier
→ surface rendering parameters
→ pointer to next surface.

• In A-buffer Algorithm scan line are processed to determine surface overlap of pixels across the individual scan lines.

• A buffer algorithm are used to view the background opaque surface through the foreground transparent surface

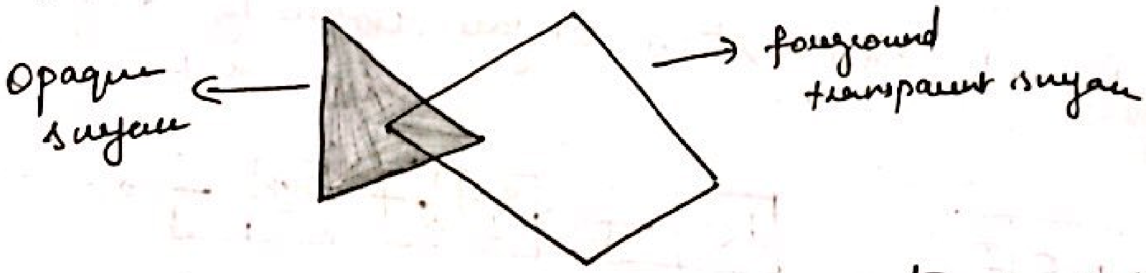


fig: Viewing an opaque surface through transparent surface

Scan line Method

* This is an image-space method used for removing hidden surfaces.
* This method is an extension of the scan-line algorithm for filling polygon interiors
* This method deals with the multiple surface instead of filling one surface.
* As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which surfaces are visible.
* Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane.
* When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.
* Three tables are set up for the various surfaces. They are
  - i) Edge Table
  - ii) polygon Table
  - iii) Active Edge table
* The edge table contains coordinate endpoints for each line in the scene, the inverse slope of each line and pointers into the polygon table to identify the surface bounded by each line.

x	Ymax	Δx	ID

x → x-coordinate of the end with the minimum y-coordinate

Y_{max} - Y-coordinate of the edges at other end point

Δx - $1/m$

ID - polygon identifier at each surface.

* The polygon table contains coefficient of the plane equation for each surface, intensity information for the surface and pointers into edge Table.

ID	Plane Coefficient	Shading information	IN/OUT

ID - Polygon surface identifier

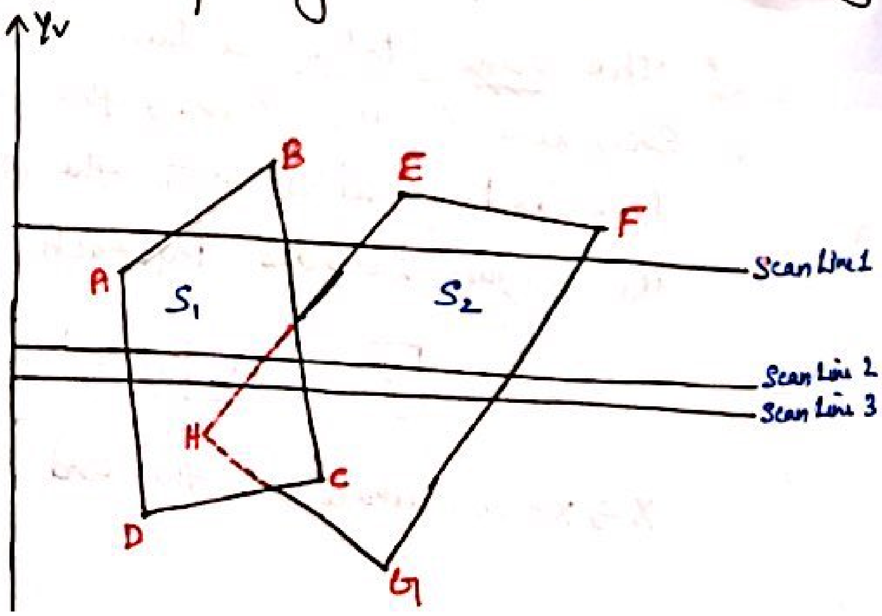
plane coefficient - Coefficients of the plane equation (A,B,C,D)

Shading Information - Intensity information of the polygon

IN/OUT flag - IN/OUT flag indicates whether a position along a scan line is inside or outside of the surface

* Active Edge Table contains list of edges that cross the current scan line, sorted in order of increasing x.

* Scan lines are processed from left to right.
* The following figure illustrates the scan-line method for locating visible portions of surface for pixel positions along the line

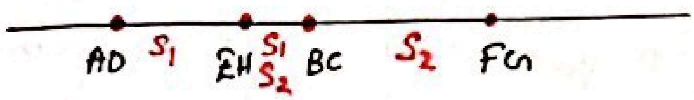


ScanLine	Edge List	Surface flag
ScanLine 1	AB, BC, EH, FG	AB - S_1 BC - S_1 EH - S_2 FG - S_2
ScanLine 2	AD, EH, BC, FG	AD - S_1 EH - S_1, S_2 BC - S_1, S_2 FG - S_2
ScanLine 3	AD, EH, BC, FG	AD - S_1 EH - S_1, S_2 BC - S_1, S_2 FG - S_2

In Scan Line 1 no Surfaces intersect with eachother, so the intensity values in the other areas are set to the background intensity

For ScanLine 2&3 the Surfaces S_1 and S_2 intersect at the edge EH and BC. In the intersection interval depth calculation must be made using the plane coefficients for the two surfaces and found the visible surface depends on the Z-value

The detailed view of the scan line 2 is given below



In the Region EH - BC, the depth calculation must be made for each pixel position.

• In the scan line method the Coherence property of the scanline and object is taken into account ~~and~~ as we pass from one Scanline to next.

with the coherence property of the scanline unnecessary depth calculation between the edges in the adjacent scanline

Disadvantages

* Scanline Method cannot be used to the surfaces which overlap through cuts or having cyclically overlap.

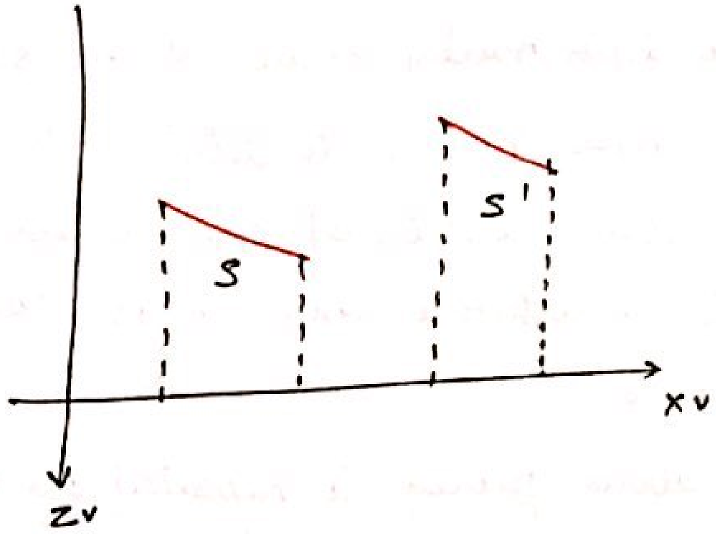
Depth Sorting Method (painters Algorithm)

* This method using both image space and object space operations
* Depth sorting methods performs the following functions
  - → Surfaces are sorted in order of decreasing depth
  - → Surfaces are scan converted in the order of the greatest depth of the surface
* Sorting operation performed on both image and object space
* Scan conversion is performed in image space method only
* Depth sorting Method is also called painters Algm because this algorithm first sort the surfaces which are far away from the view plane. At the final stage the surfaces which are near to the view plane are entered into the refresh buffer.
* Painting of polygon surfaces onto the frame buffer according to the depth is carried out in several steps.
* Assume the viewing direction is along z-axis
* Surfaces are ordered according to the largest z-value on each surface
* Surface S with the greatest depth is then compared to the other surface in the list to determine the overlaps in depth

* If no depth overlap occur, S is scan converted.
* If a depth overlap is detected at any point in the list, some additional comparison is needed to determine whether any of the surface should be reordered.
* The above process is repeated for the next surface in the list.
* Following tests are performed on each surface that overlap with 'S', if any one of the test is true, no reordering of that surface is necessary.

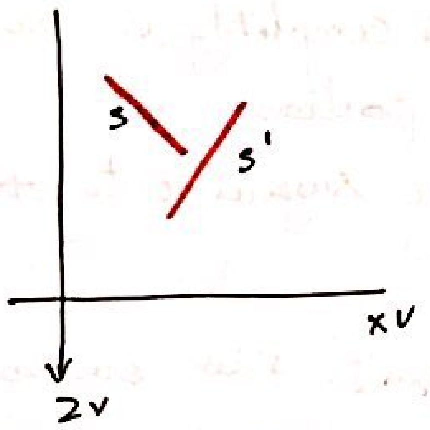
1. The bounding rectangle in the xy plane for the two surfaces do not overlap
2. Surface 'S' is completly behind the overlapping surface relative to the viewing position
3. The overlapping surface is completely in front of S relative to the viewing position
4. The projection of the two surfaces onto the view plane do not overlap.

Test 1 is performed in two parts. First overlap in the x-direction is checked then overlap for y-direction is checked. If either of these direction shows no overlap, the two plane cannot obscure (not seen) one other. The following figures shows two surfaces that overlap in z direction but not in x and y. direction

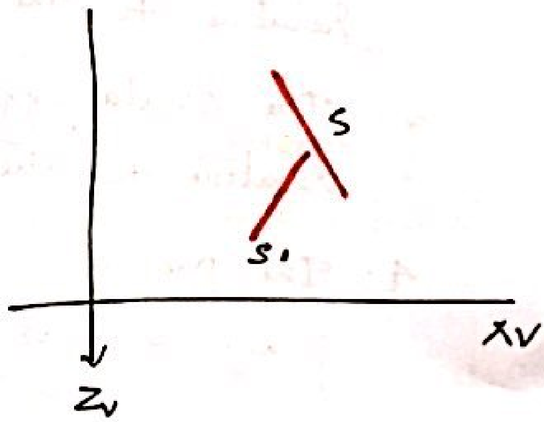


Test 2 & Test 3 can perform with an inside-outside polygon test. Initially the coordinates for all vertices of s into the plane equations for the overlapping surface and check the sign of the result.

* S is behind S' if all vertices of s are inside S'
* S is completely infront of S' if all vertices of S are outside of S'.



Surface S completely behind (inside) the overlapping surface S'



Overlapping surface S' is completely infront (outside) of surface S but S is not completely behind S'

Test 4 is performed by checking for intersection between the bounding edges of the two surface using line equations in the xy plane.

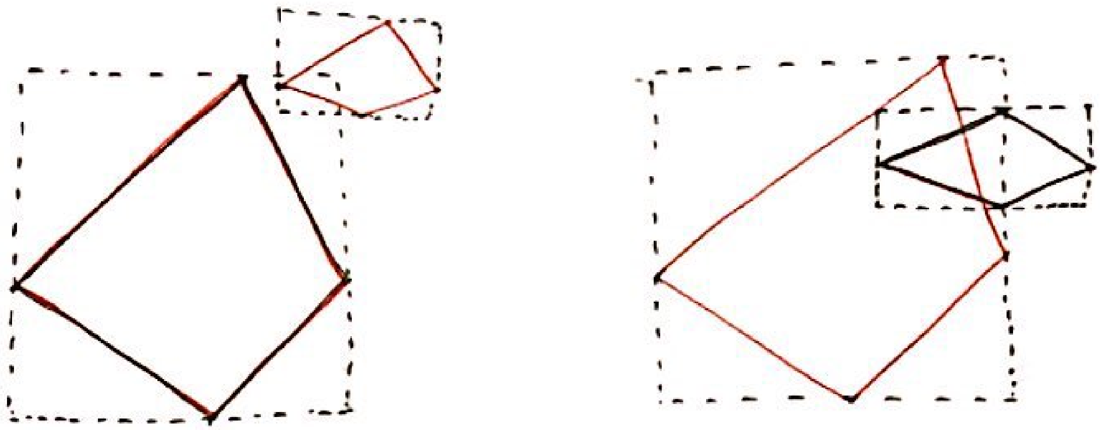


fig: Two surfaces with overlapping bounding rectangle in the xy plane.

If all the above 4 tests fails with a particular overlapping surface S', then interchange surface S and S' in the sorted list.

Disadvantage

* Algorithms get into an infinite loop
* If more than two surfaces alternately obscure with each other then algms continually reshuffle the position of the overlapping surfaces.
* Continuous reordering of the surface can be avoided by flag any surface.

Image processing

An image may be defined as a 2-D function f(x,y), where x and y are the spatial coordinates and the amplitude 'f' at any pair of coordinates (x,y) is called intensity or gray level of image at that point.

When f, x and y are finite and discrete then the image is digital image.

Digital image processing refers to the processing of digital images by means of digital computer.

Digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are called picture elements, pels or pixels. Pixel is the term most widely denote the element of a digital image.

Application of image processing

* Image sharpening and restoration
* Medical field (X ray, CT scan, PET scan, UV imaging etc)
* Remote sensing
* Transmission and encoding
* Machine/Robot vision
* Color processing
* pattern Recognition
* Video processing
* Microscopic imaging

In digital image processing the input is the image of the object and the output after digital processing is also an image with new features add to the image (intensity, resolution etc)

Fundamental steps in Digital Image processing

The fundamental steps in the digital image processing are shown in the following figure

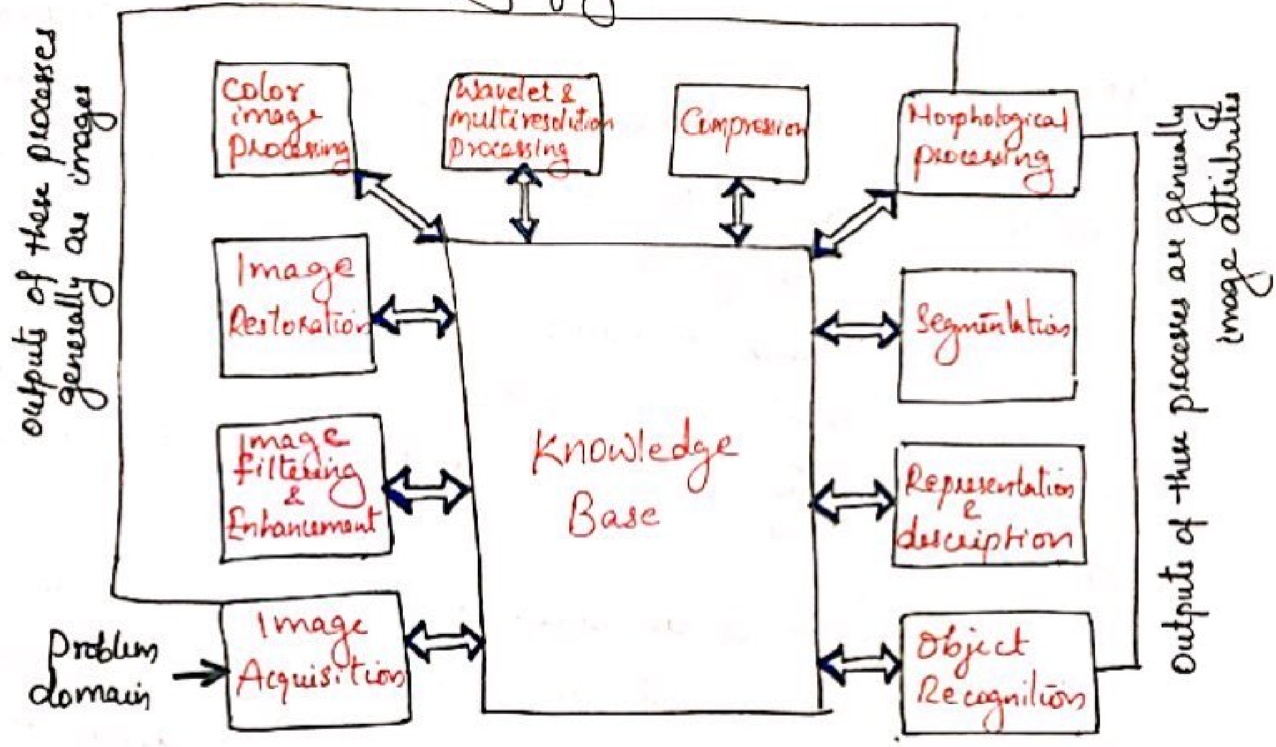


fig: Fundamental steps in digital image processing

Image Acquisition is the first process in digital image processing. In this step image is captured by a sensor and digitized. Image acquisition stage involves preprocessing such as scaling. Image acquisition process is generally achieved by suitable camera.
The aim of image acquisition is to transform an optical image into an array of numerical data.

In image acquisition different cameras are used for different applications. Film camera is used for the x-ray images, for infrared images, we use camera which are sensitive to infrared radiation. For normal images cameras which are sensitive to visual spectrum are used.

Image Enhancement is the process of manipulating an image so that the result is more suitable than the original image for the specific application. This process improves the interpretability or perception of information in images for human viewers or to provide better input for other automated image processing techniques.

Image enhancement techniques have been widely used in many applications of image processing where the subjective quality of images is important for human interpretation.

Image Restoration is an area that also deals with the improving appearance of an image. Image Restoration is ~~subj~~ objective in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.

Image Restoration is concerned with the reconstruction or estimation of the uncorrupted image from a blurred and noisy image. This process tries to perform an operation on the image that is the inverse of the imperfections in the image formation system.

Color Image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the internet. This may include color modeling and processing in a digital domain

Wavelets processing are the foundation for representing images in various degrees of resolution. This is useful for data compression and for pyramidal representation of images since the images is subdivided into the smaller regions

Compression deals with the techniques for reducing the storage required to save an image or the bandwidth required to transmit it. Image compression is minimizing the size in bytes of a graphic file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the internet or downloaded from web pages. The familiar image compression method used by most of the user are JPEG (Joint photographic Expert Group).

Morphological processing deals with the tools for extracting image components that are useful in the representation and description of shape. Morphological operations apply a structuring element to an input image to create an output image of the same size.

Image Segmentation procedures partition an image into its constituent parts or object. Autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged (uneven) segmentation procedure brings the process a long way towards successful solution of imaging problem that require objects to be identified individually. Image segmentation is an essential step in image analysis, object representation visualization & many other image processing tasks.

Representation & Description almost always follow the output of a segmentation stage, which usually is raw pixel data constituting either the boundary of a region or all points in the region itself.

Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from other.

Recognition & Interpretation : Recognition is the process that assigns label to an object based on the information provided by its description eg: vehicle.

Interpretation means assigning meaning to a recognized object.

Knowledge Base : knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated list of all major possible defects in a material inspection problem or an image database containing high resolution satellite images of a region in connection with change detection applications

Components of Image processing

The main components of image processing is shown on the following figure.

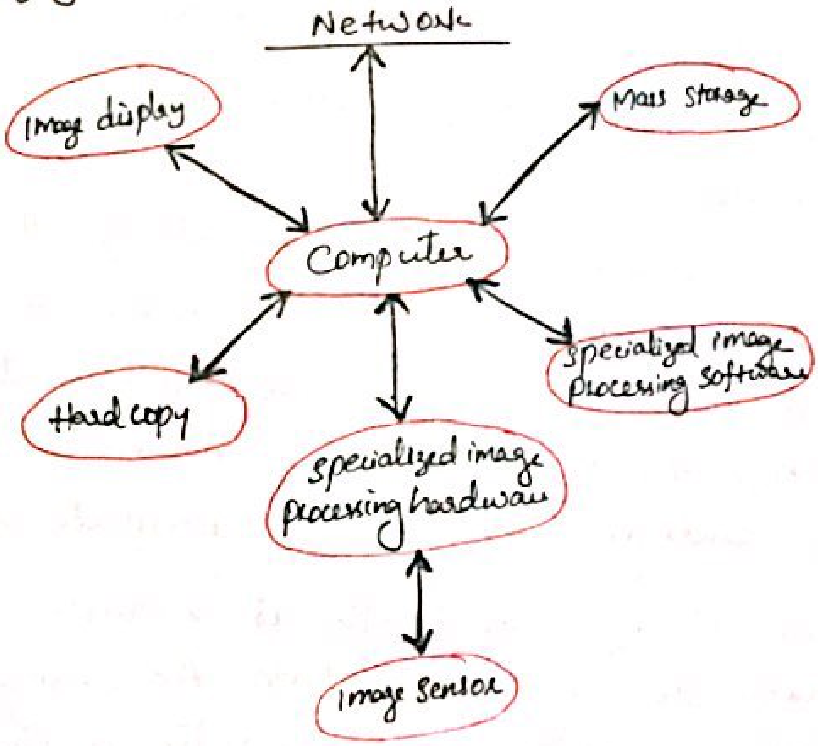


fig: components of image processing

With reference to the image sensors two elements are required to acquire digital images. The first is a physical device that is sensitive to the energy radiated by the object. The second called the digitizer which convert the output of the physical sensing device into the digital form.

Specialized Image processing hardware consists of the digitizer plus hardware that performs other primitive operations such as ALU which performs arithmetic such as addition and subtraction and logical operations in parallel on images.

Computer is a general purpose and can range from a PC to supercomputer depending on the application. In dedicated applications sometimes customise computers are used to achieve a required level of performance

Software for image processing consists of specialized modules that perform specific tasks. It includes a well defined packages that utilizes the specialized modules and includes the capability for the user to write code as a minimum. Sophisticated software package allows integration of modules and software commands from at least one computer language.

Mass storage capability is a must component in image processing application. An image of size 1024x1024 pixels, in which the intensity of each pixel is an 8-bit quantity requires one megabytes of storage space if the image is not compressed. Digital storage for image processing application falls into three categories

i) Short term storage used during processing

ii) On-line storage for relatively fast recall

iii) Archival storage such as magnetic tapes & disks

one method of short term storage is computer memory or can use a specialized boards called frame buffers that store one or more images & can access rapidly usually at video rates (30 images/sec)

Image display are usually color TV monitors. The monitors are driven by the output of images and graphic displays cards that are an integral part of computer system

Hardcopy devices are used for recording image includes laser printers, film cameras, heat sensitive devices, inkjet units and digital units such as optical & CD ROM disk. Film provides the highest possible resolution but paper is the medium of choice for written application

Networking is almost a default function in any computer systems because large amount of data inherent in image processing application. The key consideration of image transmission bandwidth. networking is in the image

Representing Digital images

Digital image representation is classified into two types.

i) Vector images

ii) Bit-map images

An image ~~of~~ can be represented by 2-D function of the form f(x,y). The value or amplitude of 'f' at spatial coordinates (x,y) is a positive scalar quantity whose meaning is determined by the source of the image.

Vector Images

One way to describe an image using numbers is to declare its contents using position and size of geometric forms and shapes like lines, curves, rectangles and circles. Such an images are called vector images.

In vector images the coordinate system is used to represent the image and the coordinate system defines elements and position in relation to each other (defines each pixel position)

The coordinate system is as shown in the following figure

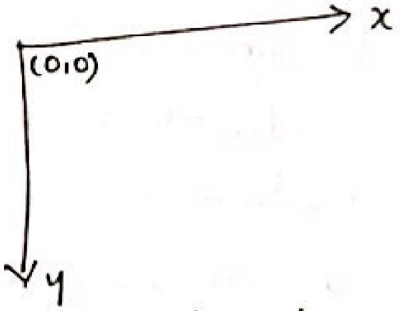


fig: Coordinate System

The image to be displayed to be translated into bitmap image and this process is called Rasterization.

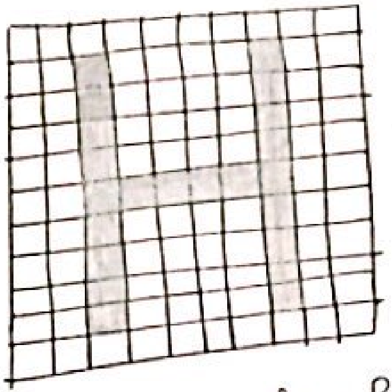
A vector image is resolution independent and the image can be enlarged or shrinked without affecting the output quality.

Vector images are the way to represent Fonts, Logos and many illustration

Bitmap images

Bitmap or Raster images are digital photographs and are the most common form to represent natural images and other forms of graphics that are rich in detail.

Bitmap images refers how graphics is stored in the video memory of a computer.
The term bitmap refers to how a given pattern of bits represents in a pixel maps to a specific color.
The bitmap image is shown in the following figure.



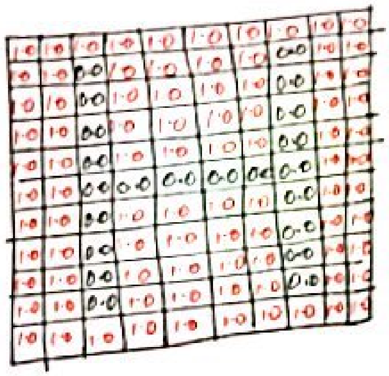


fig : Bitmap image

In the above figure, A bitmap images are the form of an array where the value of each element is called pixel picture element corresponds to the color of that portion of the image.
Each horizontal line in the image is called scan line.
In the image larger values corresponds to brighter area. while the lower value are for the dark pixels.
when measuring the value for a pixel, one takes the average color of an area around the location of the pixel.
A simplistic model is sampling a square this is called a box filter.
The number of horizontal and vertical samples in the pixel grid is called raster dimensions it is specified as width x height.

Resolution is a measurement of sampling density, Resolution of bitmap images gives a relationship between pixel dimension & physical dimension.

The process of reducing the raster dimensions is called decimation this can be done by averaging the value of source pixels contributing to each output pixel.

One of the most common pixel format used is 8bit RGB where the red, green and blue values are stored in interleaved memory.

Bitmap images occupy a lot of memory, image compression reduces the amount of memory needed to store an image. Compression ratio is the ratio between the compressed image and the uncompressed image.

There are two types of compression.

→ Lossy compression
→ Lossless compression

In Lossless compression, repetition and predictability is used to represent all the information using less memory. The original image can be restored. One of the simplest lossless image compression method is run-length encoding.

In Lossy compression method, some features of the image is lost. especially this method eliminate the redundant information. when the file is uncompressed only the part of the original information is still there.

It is generally used to compress video and sound.

where a certain amount of information loss will not be detected by most users. eg: JPEG image compression

Image Representation in 2-D

The image may be defined as a 2-D function $f(x,y)$ where x and y are spatial (plane) coordinate. The amplitude 'f' at any pair of coordinate (x,y) is called the intensity or gray level of the image at that point.

When (x,y) and amplitude values of 'f' are all finite discrete quantities, then the image is a digital image.

- Image is a 2-D function x & y.

$f(x,y)$ where $x = 0, 1, 2 \ldots N-1$
$y = 0, 1, 2 \ldots M-1$

$N \Rightarrow$ no: of pixels in x-direction
$M \rightarrow$ no: of pixels in y-direction

$f(x,y)$ can be represented as

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & & & \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Values of M & N are totally positive and cannot be a negative value.

Basic Relationships Between Pixels

Basic Relationship between the pixel can be given as

→ Neighbourhood
→ Adjacency
→ Connectivity
→ Paths
→ Regions & Boundaries

Neighbourhood

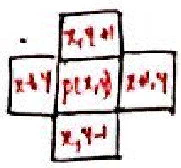
Any pixel $p(x,y)$ has two vertical and two horizontal neighbours and is given by $\{(x+1,y), (x-1,y), (x,y+1), (x,y-1)\}$

This set of pixels are known as 4 neighbours of p and is denoted by $N_4(P)$. All of them are at unit distance from p.

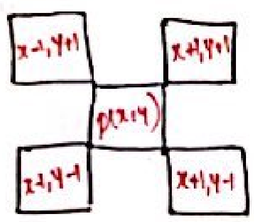
The four diagonal neighbours of $p(x,y)$ are given by $\{(x+1,y+1), (x+1,y-1), (x-1,y+1)\ (x-1,y-1)\}$. This is denoted by $N_D(P)$.

The points $N_4(P)$ and $N_D(P)$ are together known as 8-neighbours of the point p, denoted by $N_8(P)$.

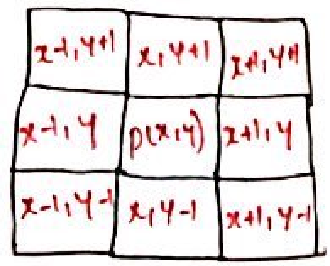
Some of the points in the N_4, N_D & N_8 may falls outside the image when p lies on the border of image



$N_4(P)$
(4 connected)



$N_D(P)$



$N_8(P)$

Adjacency

Let V be the set of intensity values used to define ~~intensity~~ adjacency. In a binary image $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a gray-scale image V contains more elements.

eg: In the adjacency of pixels with a range of intensity values 0 to 255, the set V could be any subset of these 256 values. There are 3 type of adjacency.

→ 4-adjacency: Two pixels p & q with values from V are 4-adjacent if q is in the set $N_4(p)$

→ 8-adjacency: Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

→ m-adjacency: Two pixels p and q with values from V are m-adjacent if

i) q is in $N_4(p)$ or

ii) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

Path

A path from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$ and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \le i \le n$. 'n' is the length of path.

If $(x_0, y_0) = (x_n, y_n)$ the path is closed.

4-path, 8-path and m-path can be defined on the type of adjacency specified. The following figure shows the paths between the top right and bottom right points are 8-paths

$$\begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Arrangement of pixel — pixel that are 8-adjacent — m-adjacency

Connected

Let S represent a subset of pixels in an image. Two pixels p and q are said to be connected in S if there exists a path between them consisting entirely of pixels in S. For any pixel p in S, the set of pixels that are connected to it in S is called a connected component of S. If it only has one connected component then set S is called a connected set.

Region

Let R be a subset of pixels in an image. We can call R as a region of the image if R is a connected set. Two regions R_i and R_j are said to be adjacent if their union forms a connected set. Regions that are not adjacent are said to be disjoint

Boundary

The boundary also called border or contour of a region R is the set of points that are adjacent to points in the Complement of R. The border of a Region is the set of pixels in the Region that have atleast one background neighbour.

Edge Detection

Edges are the pixels where brightness changes abruptly and that point shows sharp change in the intensity function. An edge of a image is a boundary or contour at which a significant change occurs in some physical aspect of an image, Such as the surface reflectance, illumination or the distance of the visible surface from the viewer. changes in the physical aspects can be a variety of ways including changes in the intensity, color. and texture.

Robert Detection

Robert cross gradient operators are used for 2-D mask when a diagonal edge detection is considered.

Robert edge detector is based on diagonal difference.

Consider the pixels in the z-value.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Consider our pixel of interest is z_5

z_5	z_6
z_8	z_9

Edge traversal is to z_9

The mask for the same ($z_5 - z_9$) is given as

z_5	1	0	z_6
z_8	0	-1	z_9

Consider another pixel z_6.

z_5	z_6
z_8	z_9

Edge traversal is to z_8

Mask is given as

z_5	0	1	z_6
z_8	-1	0	z_9

Robert edge detector masks are given by

0	1
-1	0

1	0
0	-1

The first operator in each pair is particularly sensitive to edges that run diagonally from the lower left of the original image to the upper right, while the second operator in each pair detects edges running from the upper left to the lower right.

2x2 mask is having problem

→ It is not easy to implement

→ No: of calculation is more

→ No: of neighbouring pixels considered in one go are less

3x3 mask of the Robert detectors are given as

-1	0	0
0	0	0
0	0	1

0	0	1
0	0	0
-1	0	0

Due to the disadvantage of Robert edge detector some changes are made to it and the changes are as follows

→ Size of the mask.

→ No: of neighbouring pixels considered

By considering the above changes two improved masks obtained

i) prewitt

ii) Sobel

Prewitt's edge detector combines uniform smoothing in one direction with the edge detection in the perpendicular direction to produce

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

These operators can be factored into the successive application of two simpler operator

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

and

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

Scanned with CamScanner

Sobel's edge detector combines binomial (1,2,1) smoothing with edge detection. It is also defined by operators that can be factored. In sobel '2' is used as centre location provides image smoothening

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

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

and

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

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

These operators can be represented as a number of shifts, additions and subtraction of the entire image. These can be performed very rapidly using suitable hardware.

Steps performed in edge Detection

1. Image Smoothing for noise reduction
2. Detection of edge points - operation that extracts from an image all points that are potential candidates to become edge points.

3: Edge Localization - The objective of this step is to select from the candidate edge points only the points that are true members of the set of points comprising an edge

Some of the techniques for achieving these steps are Robert, prewitt & sobel operator

The tool of choice for finding edge strength and direction at location (x,y) of an image f is the gradient denoted by ∇f and defined as the vector.

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude (length) of vector ∇f, denoted as $M(x,y)$ where

$$M(x,y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

is the value of the rate of change in the direction of the gradient vector.

The direction of the gradient vector is given by the angle

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$ measured with respect to x-axis

$$g_x = \frac{\partial f(x,y)}{\partial (x)} = f(x+1,y) - f(x,y)$$

$$g_y = \frac{\partial f(x,y)}{\partial (y)} = f(x,y+1) - f(x,y)$$

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object.

Types of Curves

A curve is an infinitely large set of points. Each point has two neighbors except endpoints. Curves can be broadly classified into three categories − **explicit, implicit,** and **parametric curves**.

Implicit Curves

Implicit curve representations define the set of points on a curve by employing a procedure that can test to see if a point in on the curve. Usually, an implicit curve is defined by an implicit function of the form −

$$F(x,y)=0$$

It can represent multivalue curves multiple yvalues for an xvalue multiple yvalues for an xvalue. A common example is the circle, whose implicit representation is

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

Bezier Curves

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as −

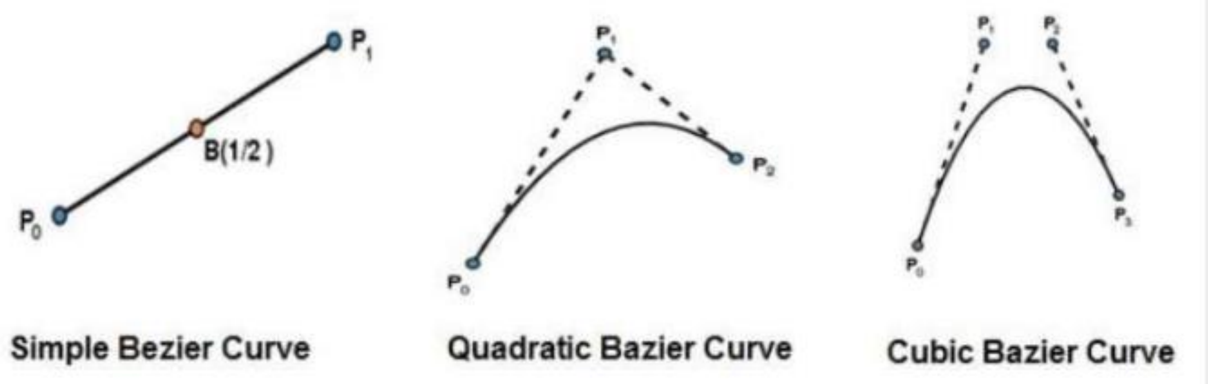
$$\sum_{k=0}^{n} P_i B_i^n(t)$$

Where p_i is the set of points and $B_{ni}(t)$represents the Bernstein polynomials which are given by –

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

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

The simplest Bézier curve is the straight line from the point P_0P0 to P_1P1. A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points.



Properties of Bezier Curves

Bezier curves have the following properties −

- They generally follow the shape of the control polygon, which consists of the segments joining the control points.
- They always pass through the first and last control points.
- They are contained in the convex hull of their defining control points.
- The degree of the polynomial defining the curve segment is one less that the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
- A Bezier curve generally follows the shape of the defining polygon.
- The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
- The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
- No straight line intersects a Bezier curve more times than it intersects its control polygon.
- They are invariant under an affine transformation.
- Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.
- A given Bezier curve can be subdivided at a point t=t0 into two Bezier segments which join together at the point corresponding to the parameter value t=t0.

B-Spline Curves

The Bezier-curve produced by the Bernstein basis function has limited flexibility.

- First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.

- The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.

The B-spline basis contains the Bernstein basis as the special case. The B-spline basis is non-global.

A B-spline curve is defined as a linear combination of control points Pi and B-spline basis function Ni, k t given by

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \qquad n \geq k-1, \qquad t \,\epsilon\, [tk-1, tn+1]$$

Where,

- {pipi: i=0, 1, 2….n} are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree k - 1,
- the Ni,k(t)Ni,k(t) are the "normalized B-spline blending functions". They are described by the order k and by a non-decreasing sequence of real numbers normally called the "knot sequence".

$$t_i{:}i=0,\ldots,\ldots n+K$$

The N_i, k functions are described as follows −

$$N_{i,1}(t) = \begin{cases} 1, & if\, u \,\epsilon\, [t_i, t_{i+1}) \\ 0, & Otherwise \end{cases}$$

and if k > 1,

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1}} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

and

$$t \,\epsilon\, [t_{k-1}, t_{n+1})$$

Properties of B-spline Curve

B-spline curves have the following properties −

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

Visual Perception

Visual perception is the ability to perceive our surroundings through the light that enters our eyes. The visual perception of colors, patterns, and structures has been of particular interest in relation to graphical user interfaces (GUIs) because these are perceived *exclusively* through vision. An understanding of visual perception therefore enables designers to create more effective user interfaces.

Physiologically, visual perception happens when the eye focuses light on the *retina*. Within the retina, there is a layer of photoreceptor (light-receiving) cells which are designed to change light into a series of *electrochemical signals* to be transmitted to the brain. Visual perception occurs in the brain's *cerebral cortex*; the electrochemical signals get there by traveling through the optic nerve and the thalamus. The process can take a mere 13 milliseconds, according to a 2017 study at MIT in the United States.

Different attributes of visual perception are widely used in GUI design. Many designers apply Gestalt principles (i.e., how humans structure visual stimuli) to the design of GUIs so as to create interfaces that are easy for users to perceive and understand. The visual perception of affordances (action possibilities in the environment) is another example of how the understanding of visual perception is a critical item in any designer's toolkit.