Computer Graphics Copy Notes

ICT 6th Semester 2076

by

Mukesh Singh

ICT 5th batch, 2073

Sukuna Multiple Campus
Sundarharaincha-12, Morang

Unit-1

~~History Of~~ Introduction

1.1. History of Computer graphics

Definitions

Computer graphics is the discipline of generating images with the aid of computers.

Computer graphics is an art of drawing pictures, lines, charts, etc using computers with the help of programming.

History

Computer Graphics (CG) was first created ~~by~~ as a visualization tool for scientists and engineers in government and corporate research centers such as Bell Labs and Boeing in the 1950s.

1950s

The true beginning of computer graphics comes from the SAGE computer system, which was designed to support military preparedness.

Development of Whirlwind Computer in 1950.

1960s

Computer graphics really emerged during the 1960s. At the time, anti-aliased lines, circles and curved drawings, which we now consider rudimentary operations, were major topics in Computer graphics.

1970s

Rendering and (shading) were discovered by Gouraud and Phong at the University of Utah.

Keyframe-based animation for 3D graphics was demonstrated for the first time.

Xerox ~~Park~~PARC developed a "paint" program.

Arcade games, such as Pong and Pac-Man, became extremely popular.

1980s

Near the mid-1980s, Jim Blinn introduced blobby model and texture mapping concepts.

Computer Aided Design courses began to be taught at various universities.

Adobe brought its photoshop software to the market.

Video games took off.

1990s

In 1992, the OpenGL became the standard for graphics APIs.

The first CAD College was opened.

PC graphics cards, like 3dfx and Nvidia were introduced.

Graphics effects in Movies such as Terminator 2, Jurassic Park and Toy Story, became omnipresent.

2000s

As we moved into the 21st century, onlin CAD courses became available, blending design and technology to create new frontiers in architecture, engineering and much more.

1.2 Applications of Computer Graphics

Some of the applications of Computer graphics are explained below:

1. Computer Art → Computer graphics are used in the field of commercial arts which include animation packages, paint packages. It is used to generate television and advertising commercial. Cartoon drawing, paintings, logo design can also be done.

2. Computer Aided Drawing → Design of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details of the drawing and producing more accurate and sharp drawings with better specifications.

3. Presentation Graphics → Example of presentation graphics are bar graphs, pie charts, time charts showing the relationships between multiple parameters. Presentation graphics is commonly used to summarize: financial, statistical, mathematical, scientific, managerial and other types of reports.

4. Entertainment → Computer graphics are now commonly used in making motion pictures, music videos and television shows. It is also used in making games.

5. Education → Using computer graphics, many educational models can be created through which more interest can be generated among the students regarding the subject.

6. Training → Specialized systems for training like simulators can be used for training the candidates in a way that can be grasped in a short period of time with better understanding. Creation of training modules using computer graphics is simple and very useful.

7. Visualization → It is used for visualization of scientists, engineers, medical personnel, business analysts for the study of a large amount of information.

8. Image Processing → Processing of existing images into refined ones for better interpretation is one of the many applications of computer graphics.

9. Printing Technology → Computer graphics is used for printing technology and textile design.

10. Educational Software → Computer graphics is used in the development of educational software for making computer-aided instruction.

1.3. Picture representation

Pictures are represented as a collection of discrete picture element called Pixels. It is the smallest addressable screen element.

The process of determining the appropriate pixels for representing pictures or objects is called Rasterization.

Rasterization is the task of taking an image described in a vector graphics format (shapes) and converting it into a raster image (pixels or dots) for output on a video display or printer, or for storage in a bitmap file format.

1.4. Properties of light

Properties of light are discussed below:

1. Effects of Materials on Light

Materials can be classified based on how it responds to light incident on them:

i) Opaque materials → absorb light; do not let light to pass through.

ii) Transparent materials → allow light to easily pass through them.

iii) Translucent materials → allow light to pass through but distorts the light during the passage.

2. Reflection

Light follows the law of reflection: "The angle of incidence is equal to the angle of reflection."

Light can bounce off materials in two ways:

i) Diffuse reflection → reflected rays goes in different directions; happens in rough textured or uneven surfaces.

ii) Regular reflection → reflected rays goes in one direction; happens in smooth and shiny surfaces; image can be seen.

3. Refraction

Refraction is a change on the direction of light when it passes from a medium to another one.

The "law of refraction" states that, "the incident ray, the refracted ray, and the normal to the interface all lie in the same plane."

Light travels faster in air, slow in water and slower

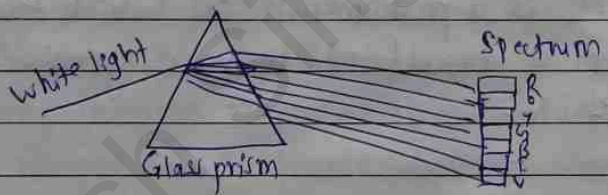
still in a glass.

The measure of how much light refracts in a medium is called index of refraction.

Medium	Index of refraction (n)
air	1.000293
water	1.3330
glass	1.490
diamond	2.419

4. Dispersion

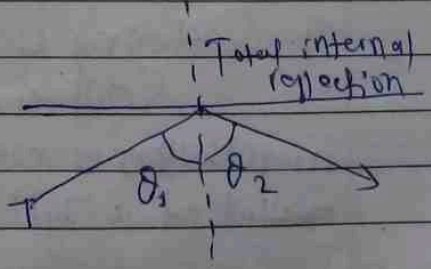
The dispersion of light is the phenomenon of splitting of a beam of white light into its seven constituent colours when passed through a transparent medium. It was discovered by Sir Isaac Newton in 1666.



5. Total Internal Reflection

TIR is the complete reflection of a ray of light within a medium such as water or glass from the surrounding surfaces back into the medium. The phenomenon occurs if the angle of incidence is greater than a certain limiting angle, called the critical angle.

6. ~~Diffraction~~



6. Diffraction

Diffraction is the slight bending of light as it passes around the edge of an object.

The amount of bending depends on the relative size of the wavelength of the light compared with the size of the opening.

7. Interference

When two light waves from different coherent sources meet together, then the distribution of energy due to one wave is disturbed by the other. This modification in the distribution of light energy due to super position of two light waves is called "Interference of light".

It is a phenomenon in which two waves superpose to form a resultant wave of greater, lower or the same amplitude.

8. Polarization

A light wave that is vibrating in more than one plane is referred to as unpolarized light.

Polarized light waves are light waves in which the vibrations occur in a single plane.

The process of transforming unpolarized light into polarized light is known as polarization.

9. Scattering of light

Scattering involves light being absorbed by irregularities or particles on a medium, then being emitted at a slightly different wavelength.

1.5. Color Models

A color model is a system for creating a full range of colours from a small set of primary colors.

There are two types of color models:

i) Additive

ii) Subtractive

i) Additive Color Model (RGB)

Additive color models use light to display color. Colors perceived in the additive models are the result of transmitted light.

The RGB color model is an additive color model. In this case R(red), G(green) and B(blue) light are added together in various combinations to reproduce a wide spectrum of colors.

i.e. Red + Green + Blue = White

The primary purpose of the RGB color model is for the display of images in electronic systems, such as on television screens and computer monitors and it's also used in digital photography.

CRT, LCD, LED and Plasma displays all utilize the RGB model.

ii) Subtractive Color Model (CMYK)

CMYK (subtractive) color model is the standard color model used in offset printing for full-color documents. Colors perceived in subtractive models are the result of reflected light.

Such printing uses inks of these four basic colors, it is also called four-color printing.

When two colors of RGB overlaps, we see a new

Color formed by mixing of the two additive primaries. These new colors are:

- A greenish blue called Cyan.
- A bluish red called magenta
- A bright Yellow
- The key color, Black (K is used instead of B)

Using (+) to mean additive mixing of ~~color~~ light and (-) to mean subtraction of light, we have:

- C (Cyan) = G + B = W - R
- M (Magenta) = R + B = W - G
- Y (Yellow) = R + G = W - B

1.6. Graphic Standards and Graphical file formats

There are a number of different types of graphics file formats. Each type stores a graphics data in a different way. Bitmap, vector and metafile formats are by far the most commonly used formats, and we focus on these.

However, there are other types of formats as well - scene, animation, multimedia, hybrid, hypertext, hypermedia, 3D, virtual modelling, reality language (VRML) etc

1. Bitmap formats

Bitmap formats are used to store bitmap data. files of this type are particularly well-suited for the storage of real-world images such as photographs and video images. Bitmap files, sometimes called raster files, essentially contain an exact pixel-by-pixel map of an image.

Microsoft BMP, PCX, TIFF and TGA are examples of commonly used bitmap formats.

2. Vector formats

Vector formats are files are particularly useful for storing line-based elements, such as lines and polygons, or those that can be decomposed into simple geometric objects, such as text. Vector files contains mathematical descriptions of image elements, rather than pixel values.

AutoCAD DXF and Microsoft SYLK are examples of commonly used vector formats.

3. Metafile formats

Metafiles can contain both bitmap and vector data in a single file. Metafiles are frequently used to transport bitmap or vector data between hardware platforms or to move image data between software platforms.

WPG, Macintosh PICT, and CGM are examples of commonly used metafile formats.

4. Scene formats

Scene format files are designed to store a condensed representation of an image or scene, which is used by a program to reconstruct the actual image.

Vector files contain description of portions of an image whereas scene files contain instructions that the rendering program uses to construct the image.

5 Animation formats

Animation formats have been around for some time. Very primitive animation formats store entire images that are displayed in sequence, usually in a loop. Advanced animation formats store only ~~the pixels that have actually changed as each frame~~ the differences between two adjacent images (called frames) and update only the pixels that have actually changed as each frame is displayed.

A display rate of 10-15 frames per second is typical for cartoon-like animations. Video animations usually require a display rate of 20 frames per second or better.

TDDD and ITDDD are examples of animation formats.

6. Multimedia formats

Multimedia formats are relatively new but are becoming more and more important. They are designed to allow the storage of data of different types in the same file. Multimedia file usually allow the inclusion of graphics, audio, and video information.

Microsoft's RIFF, Apple's QuickTime, MPEG, and Autodesk's FLI are well-known examples.

Unit-2
Hardware Concepts

2.1 Input Hardware: Keyboards, Mouse, Trackball, Joysticks Data Globe, Digitizer, Scanner, Touch Panels, Light Pens, Voice System.

Input hardware or input device is used to feed data or information into a computer system. It is generally used to provide input to the computer so that output is generated. Data input device like keyboard is used to provide additional data to the computer whereas pointing and selection devices like mouse, light-pens, touch pads are used to provide visual and indication input to the application.

1. Keyboard

An alphanumeric keyboard is used for entering text in a graphics system. It is an efficient device for inputting non-graphic data as picture labels associated with a graphics display.

Cursor-control keys and function keys are the most common features of general purpose keyboards. Function keys are used for entering commonly used operations in a single key stroke and cursor-control keys are used to select displayed objects or coordinate positions by positioning the screen cursor.

2. Mouse

Mouse is a serial input device used to select object movement in graphics system. It converts horizontal movement into vertical movement.

According to working mechanism there are 2 types of mouse.

a) Mechanical Mouse → Mechanical mouse contains two rollers.

One counts horizontal movement whereas other roller counts vertical movement. The motion of the roller in the base of mechanical mouse is converted to digital values that are used to determine the amount and direction of movement.

b) Optical Mouse → Optical mouse uses a ~~grid of~~ special mouse pad with a matrix of black and white grid of lines. The matrix reflects ~~the~~ light from LED on the bottom of the mouse. It is used for making relative changes in the position of the screen cursor.

3. Trackball

It is a pointing device. It is similar to a mouse. This is mainly used in notebook or laptop computer instead of a mouse. This is a ball which is half inverted, and by changing fingers on the ball, the pointer can be moved.

4. Joysticks

A Joystick is also a pointing device which is used to change cursor position on a monitor screen. Joystick is a stick having a spherical ball as its both lower and upper ends. The lower spherical ball moves in a socket. The Joystick can be changed in all four directions. It is mainly used in CAD and ~~play~~ playing computer games.

5. Data glove

A ~~data~~ Data Glove is ~~a~~ like a glove like device which is worn on the hand to provide inputs to virtual Reality systems and robotics.

The data glove is embedded with number of sensors

to provide the inputs to the systems based on the movements of the hand which are interpreted by softwares accompanied with the Data Glove. It may also contain the magnetic tracking device to track the global position of the glove.

6. Digitizer

Digitizer is an input device which converts analog information into digital form. Digitizer can convert a signal from the television or camera into a series of numbers that could be stored in a computer. They can be used by the computer to create a picture of whatever the camera had been pointed at.

Digitizer is also known as Tablet or Graphics Tablet as it converts graphics and pictorial data into binary inputs. A graphics tablet as digitizer is used for fine works of drawing and image manipulation applications.

7. Scanner

Scanner is an input device, which works more like a photocopy machine. It is used when some information is available on paper and it is to be transferred to the hard disk of the computer for further manipulation.

Scanner captures images from the source which are then converted into a digital form that can be stored on the disk. P

8. Touch Panels

A touch panel is a piece of equipment that lets users interact with a computer by touching the screen directly.

Touch panel is a type of ~~screen~~ display screen that

has a touch-sensitive transparent panel covering the screen. A touch panel registers input when a finger or other object comes in contact with the screen.

9. Light Pens

Light pen (similar to the pen) is a light-sensitive input pointing device, basically a stylus, which is used to select a displayed menu item or drawn pictures on the monitor screen.

When its tip is moved over the monitor screen, and pen button is pressed, its photocell sensing element detects the screen location and sends the corresponding signals to the CPU.

10. Voice Systems

Voice system (voice recognition) is one of the newest, most complex input techniques used to interact with the computer. The user inputs data by speaking into a microphone.

Voice recognition is used in some graphics workstations as input devices to accept voice commands. The voice-system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases.

2.2. Video Display Devices: Refresh Cathode-Ray Tubes, Raster Scan display architecture, Random Scan display architecture, Flat-Panel displays

The most commonly used display device is a video monitor. The operation of most video monitors is based on CRT (Cathode-Ray Tube).
The following display devices are used:

1. Refresh Cathode-Ray Tube

CRT stands for Cathode Ray Tube.
A CRT is a specialized vacuum tube in which images are produced when an electron beam strikes a phosphorescent surface. It modulates, accelerates and deflects electron beam(s) onto the screen to create the images.
CRT is a technology used in traditional computer monitors and televisions.

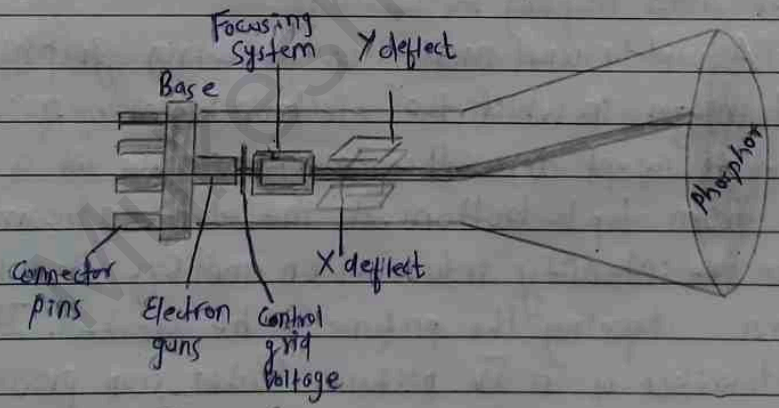


Fig: CRT

Components of CRT are:
i) Electron Gun → It consists of a series of elements, primarily a heating filament (heater) and a cathode. The electron gun creates a source of electrons which are focused into a narrow beam directed at the face of the CRT.

ii) Control Electrode → It is used to turn the electron beam on and off.

iii) Focusing System → It is used to create a clear picture by focusing the electrons into a narrow beam.

iv) Deflecting Yoke → It is used to control the direction of the electron beam. It creates an electronic or magnetic field which will bend the electron beam as it passes through the area.

v) Phosphorous Coated Screen → The inside front surface of every CRT is coated with phosphorous. Phosphorescense is the term used to characterize the light given off by a phosphorous after it has been exposed to an elatron beam.

2. Raster Scan Display Architecture

A widely used method of presenting graphical or pictorial images in which the electron beam of a cathode-ray tube is swept across the screen, one row at a time and from top to bottom. As the electron beam sweeps, the beam intensity is turned on and off dependent on information defining the picture to be created. It is similar to the definition of a TV picture. Raster scan provides a refresh rate of 60 to 80 frames per second.

At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line as shown in the figure next.

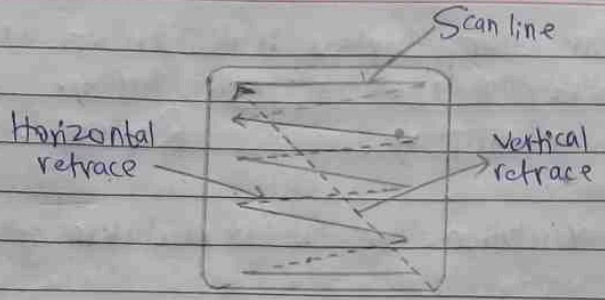
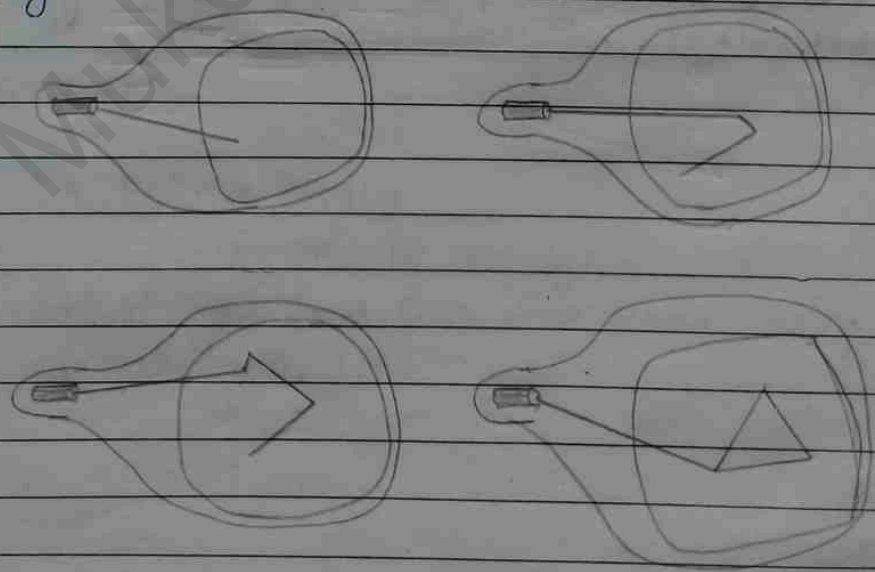


Fig: Raster Scan Display Architecture

3. Random Scan display architecture

Random Scan ~~System~~ display system uses an electron beam which operates like a pencil to create a line image on the CRT screen. The picture is constructed out of a sequence of straight-line segments. Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where its x and y coordinates define each point. After drawing the picture, the system cycles back to the first line and design all the lines of the image 30 to 60 times each second. The process is shown in the figure:



Page No.
Date

Random scan display is also known as vector display or Stroke writing display or Calligraphic display.

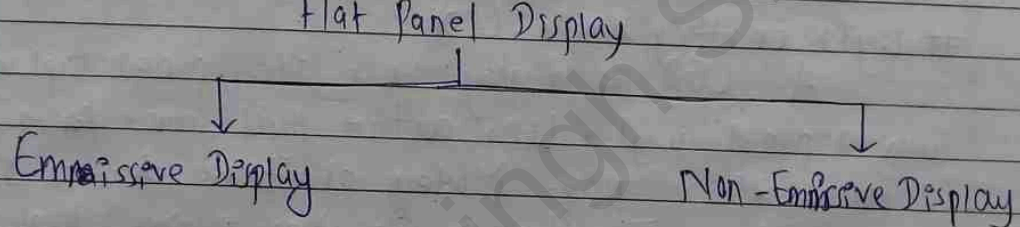
Random Scan display	Raster Scan display
1. It has high resolution.	1. Its resolution is low.
2. It is more expensive.	2. It is less expensive.
3. Modification is easy.	3. Modification is tough.
4. Refresh rate depends on resolution.	4. Refresh rate does not depends on resolution.
5. Only screen view on an area is displayed.	5. Whole screen is scanned.
6. Beam penetration technology comes under it.	6. Shadow mark technology comes under it.
7. It does not use interlacing method.	It uses interlacing.

4. Flat-Panel displays

A flat-panel display is a television, monitor or other display appliance that uses a thin panel design instead of traditional cathode ray tube (CRT) design.

They are much lighter, thinner, less power consumption and portable than CRT. They also have high resolution than older models.

Examples :→ Small T.V. monitor, calculator, pocket video games, laptop computers etc.



Flat panel displays are of two types:

1. Emmissive Display

The emissive displays are devices that convert electrical energy into light.

Examples are- Plasma Panel, thin film electroluminescent display and LED.

2. Non-Emissive Display

The non-emissive displays use optical effects to convert sunlight or light from some other source into graphics patterns.

Examples - LCD, LED

Plasma Panel Display

Plasma-panels are also called as Gas Discharge Display. It consist of an array of small lights. Lights are fluorescent in nature.

A plasma display is a type of flat panel display that uses plasma, an electrically charged ionized gas, to illuminate each pixel in order to produce a display output.

Components of Plasma-panel display are:
i) Cathode ii) Anode iii) Fluorescent Cell iv) Gas plates

Light Emitting Diode (LED)

A LED display is a flat panel display that uses a panel (array) of LEDs as the light source for or pixels for a video display. Modern electronic devices such as mobile phones, TVs, tablets, computer monitors, laptop screen etc. use a LED display to display their output.

Liquid Crystal Display (LCD)

A liquid crystal display is a flat panel display that makes use of liquid crystals that open or close when stimulated by an electric current.

2.3. Hard-Copy Devices

It is an electromechanical device, which accepts data from a computer and translates them into human understandable form.

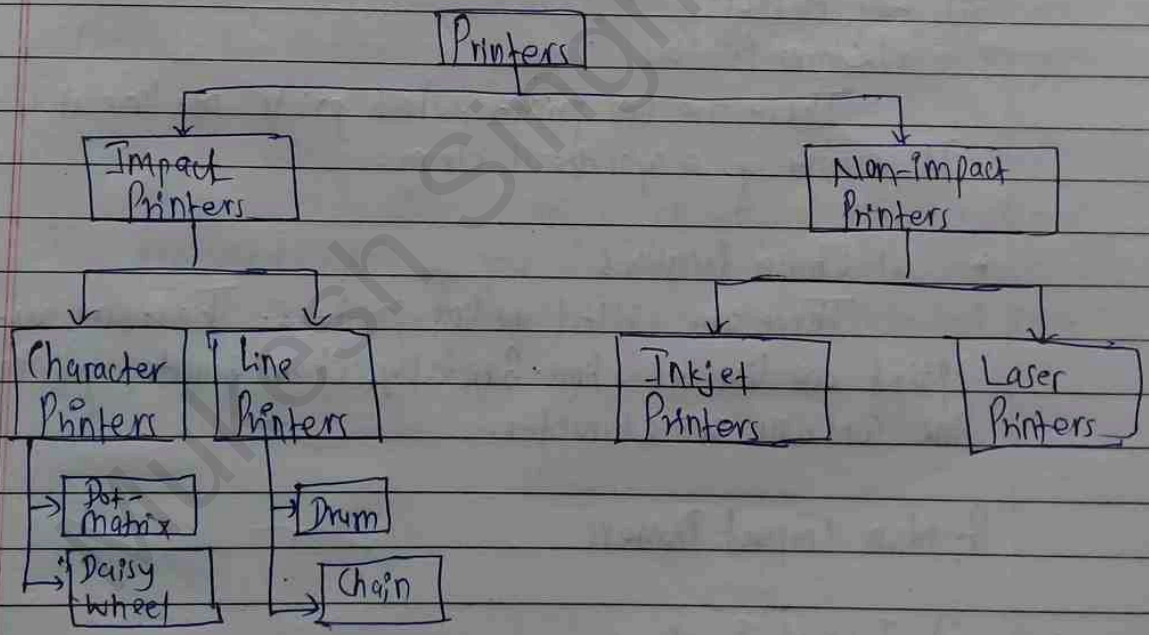
Following are Hard-Copy devices:

1. Printers
2. Plotters

1. Printers

Printer is the most important output device which is used to print hard copy output data on the paper.

Printers are of following Types:



A. Impact Printers

The printers that print the character by striking against the ribbon and onto the papers are known as Impact Printers.

There are of two types:

i) Character Printer

a) Dot Matrix Printers

Dot matrix printer has printed in the form of dots. It can print any shape of the character. This allows printer to print special character, charts, graphs etc.

b) Daisy Wheel Printers

Head is lying on a wheel and pins corresponding to character like petals of Daisy, that's why called Daisy Wheel printer. B. They are more reliable and better than Dot matrix printers.

ii) Line Printers

a) Drum Printers

These are line printers, which prints one line at a time. It consists of a cylindrical drum.

b) Chain Printers

These are called as line printers. These are used to print one line at a time. Basically, chain consists of links. Each link contains one character.

B. Non-Impact Printers

i) Inkjet Printers

These are non-impact printers which print text and images by spraying tiny droplets of liquid onto the paper. They have low cost, high quality, easy to use.

ii) Layer Printers

These are non-impact printers. They use laser lights

to produce the dots needed to form the characters to be printed on a page & hence the name laser printers.

2. Plotters

Plotters are output devices used to produce single or multiple coloured images and drawings. They use ink pen or ink-jet for drawing. It can produce high quality output on large sheets.

i) Drum Plotters

It consists of a drum. Paper on which design is made is kept on the drum. The graph plotting program controls the movement of pen and drum.

ii) Flatbed Plotter

It is used to draw complex design and graphs, charts. The plotter consists of pen and holder.

It is used to draw: cars, ships, airplanes, shoe and dress designing, road and highway design etc.

Unit-3

Output Primitives

Output primitives are basic geometric structures such as straight line segments (pixel array) and polygon colour areas, used to describe the shapes and colors of the objects. Points and straight line segments are the simplest geometric components of pictures. Additional output primitive include : circles and other conic sections, quadric surfaces, spline curves and surfaces, polygon color areas and character strings.

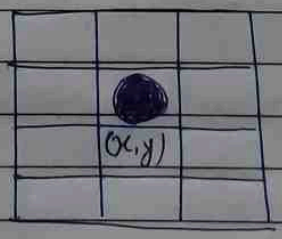
3.1. Points and Lines

Points

A geometrical point is a location in space. It has no other characteristics. It has no length, width or thickness. It is pure location.

Point is a most basic graphical element of picture representation and is completely defined by a pair of user coordinates (x, y).

A point is shown by illuminating a pixel on the screen :

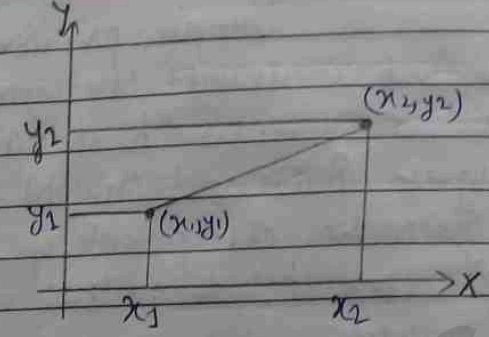


Lines

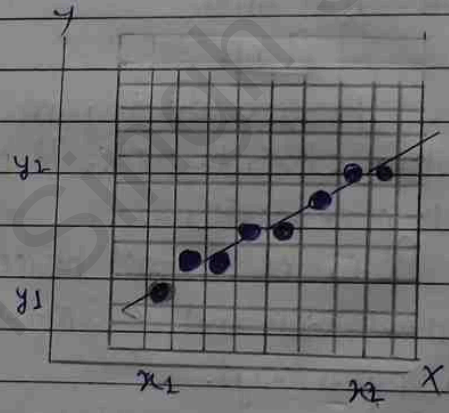
A line segment is the straight path between two points. It is the part of a line between two endpoints. It has no thickness. There is only one line between two points.

A line segment is thus defined as

$$\text{lineseg} = \{(x_1, y_1), (x_2, y_2)\}$$



. A line is produced by means of illuminating a set of intermediary pixels between two endpoints.



2.2. Line Drawing Algorithms : DDA Algorithms, Bresenhams Line Drawing Algorithm.

Line Drawing Algorithms

The process of 'turning on' the pixels for a line segment is called vector generation or line generation, and the algorithms for them are known as vector generation, algorithm or line drawing algorithms.

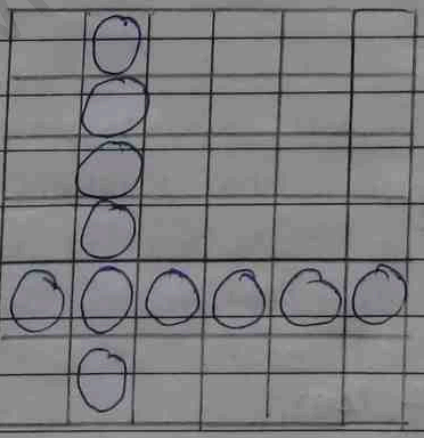
A line drawing algorithm is a graphical algorithm for approximating a line segment on discrete graphical media.

Before discussing specific line drawing algorithms it is useful to note the general requirements for such algorithms. These requirements specify the desired characteristics of line:

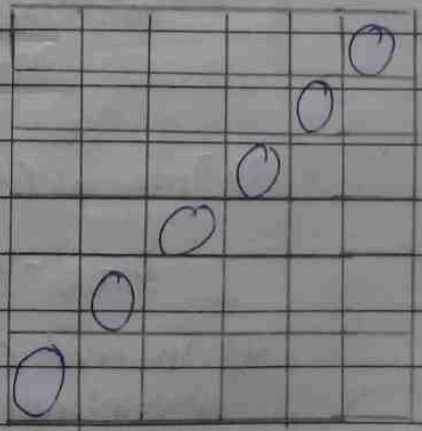
→

- The line should appear as a straight line and it should start and end accurately.
- The line should be displayed with constant brightness along its length independent of its length and orientation.
- The line should be drawn rapidly.

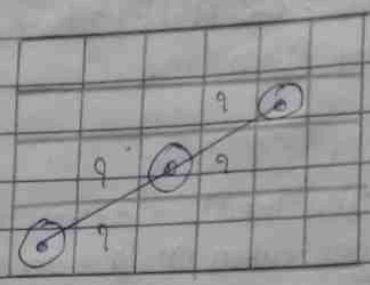
Let us see the different lines drawn in fig below:

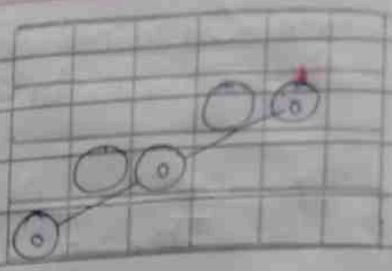


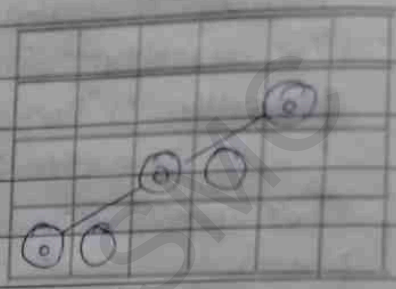
Vertical & Horizontal lines



45° line



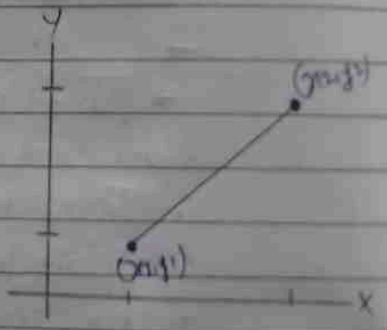




Line with other orientations

Some fundamental Equations

All line drawing algorithms make use of fundamental equations:



1) Line equation / Slope-intercept equation:

$$y = mx + b$$

2) Slope

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

3) Intercept (or y-intercept)

$$b = y - mx_1$$

4) Interval Calculation:

a) x-interval $\rightarrow \Delta x = \Delta y / m$

b) y-interval $\rightarrow \Delta y = m \cdot \Delta x$

3.2.1. DDA Algorithm

DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. In this method, calculation is performed at each step but by using results of previous steps.

It is a line algorithm based on calculating either Δy or Δx using the above equations;

$$\Delta y = m . \Delta x.$$

There are two cases:

1. DDA with Positive Slope
2. DDA with Negative Slope

1. DDA with Positive Slope

Consider a line with positive slope.

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

i) For $m <= 1$ $(|\Delta y| < |\Delta x|)$,

Sample line at unit interval in X co-ordinate.

Take $\Delta x = 1$ and Compute successive y by:

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

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

Here, k takes the value from starting point and increase by 1 until final end point. 'm' can be any real value between 0 and 1.

2. DDA with Negative Slope

ii) for $m > 1$ $(|\Delta y| > |\Delta x|)$

- Sample line at unit interval in y co-ordinate.

Take $\Delta y = 1$ and Compute successive 'x' from

$x_{k+1} = x_k + 1/m$ $\quad \boxed{\Delta y = y_{k+1} - y_k = 1}$

These above equations are under the assumptions that the line are processed from left to right endpoint.

In case, if the line is processed from ~~right to~~ Right endpoint to Left endpoint, then

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

$y_{k+1} = y_k - m$ for $m \leq 1$

And

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

$x_{k+1} = x_k - 1/m$ for $m > 1$

DDA Line Algorithm Steps

Step 1: Start Algorithm

Step 2: Read the input of the 2 endpoints of the line as (x_1, y_1) & (x_2, y_2) such that $x_1 != x_2$ and $y_1 != y_2$.

Step 3: Declare $x_1, y_1, x_2, y_2, dx, dy, x, y$ as integer variables. (here, $\Delta x = dx$ and $\Delta y = dy$)

Step 4: Calculate $dx = x_2 - x_1$ and $dy = y_2 - y_1$

Step 5: if (abs(dx) >= abs(dy)) [Note: abs = absolute]
    Step = abs(dx)
  else
    Step = abs(dy)

Step 6: Xinc = dx/step & [Note: Inc = increment]
    Yinc = dy/step
    assign x = x1
Step 7: assign y = y1

Step 7: Set pixel (x, y)

Step 8: x = x + Xinc
    y = y + Yinc
    Set pixels (Round(x) and Round(y))

Step 9: Repeat step 8 until x = x2
Step 10: End algorithm.

Example
1. Calculate the points between the starting point (5, 6) and ending point (8, 12).
Soln: Given
P(5, 6) = (x1, y1)
(8, 12) = (x2, y2)

Calculating Δx, Δy and m
- Δx = x2 - x1 = 8 - 5 = 3
- Δy = y2 - y1 = 12 - 6 = 6
- m = Δy / Δx = 6/3 = 2

Calculating number of steps

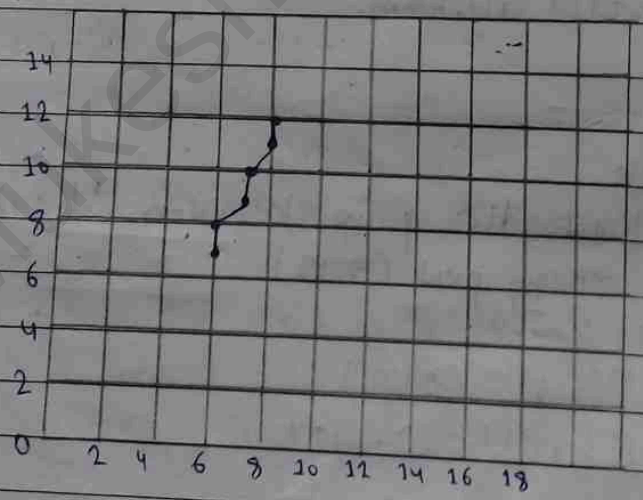
As, $|\Delta x| < |\Delta y| = 3 < 6$, so number of steps $= \Delta y = 6$

As $m > 1$, so $x_{k+1} = \text{round off}(x_k + 1/m)$

$y_{k+1} = \text{round off}(y_k + 1)$ or $(1 + y_k)$

Now,

x_k	y_k	x_{k+1}	y_{k+1}	Round off (x_{k+1}, y_{k+1})
5	6	5.5	7	(6, 7)
		6	8	(6, 8)
		6.5	9	(7, 9)
		7	10	(7, 10)
		7.5	11	(8, 11)
		8	12	(8, 12)



Q2. Calculate the points between the starting point (1,7) and ending point (11,17).

Soln:

Given

Starting coordinate $(x_1, y_1) = (1,7)$

Ending Coordinate $(x_2, y_2) = (11,17)$

Calculating Δy, Δx and m

$\Delta x = x_2 - x_1 = 11 - 1 = 10$

$\Delta y = y_2 - y_1 = 17 - 7 = 10$

$m = \Delta y / \Delta x = 10/10 = 1$

Calculating the no. of steps

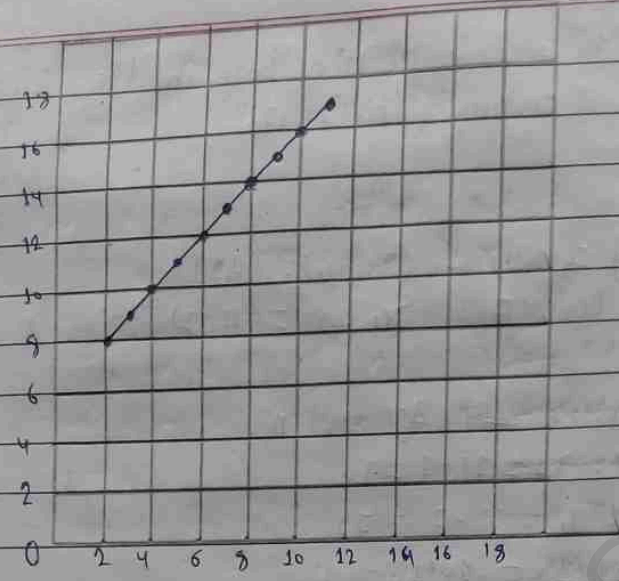
As; $|\Delta x| = |\Delta y| = 10$, So no. of steps $\Delta = \Delta y = \Delta x = 10$

As $m = 1$, so. $x_{k+1} =$ round off $(x_k + 1)$ or $(1 + x_k)$

$y_{k+1} =$ round off $(y_k + 1)$ or $(1 + y_k)$ or $(y_k + m)$

Now,

x_k	y_k	$x_k + 1$	$y_k + 1$	Round off $(x_{k+1}, y_k + 1)$
1	7	2	8	(2, 8)
		3	9	(3, 9)
		4	10	(4, 10)
		5	11	(5, 11)
		6	12	(6, 12)
		7	13	(7, 13)
		8	14	(8, 14)
		9	15	(9, 15)
		10	16	(10, 16)
		11	17	(11, 17)



C Program to for DDA Line Drawing Algorithm:
Using While

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
  void main()#include <math.h>
  void main()
  {
  int gd=DETECT, gm, i, x1, y1, x2, y2, steps;
  float x, y, dx, dy, xinc, yinc;
  initgraph(&gd, &gm, "C:\\turboc\\bgi");

  printf("Enter the starting points x1 & x2 \n");
  scanf("%d %d", &x1, &y1);
  printf("Enter the end points x2 & y2 \n");
  scanf("%d %d", &x2, &y2);
  dx=x2-x1;  dx=abs(x2-x1);
  dy=y2-y1;  dy=abs(y2-y1);
```

```
if (abs(dx) >= abs(dy))
    step = dx;
else
 step = dy;

xinc = dx/(float)step;
yinc = dy/(float)step;
x = x1;
y = y1;

putpixel(x,y,0); putpixel(x1,y1,4);
for(i=0; i<step; i++)
{
 
x = x+xinc;
y = y+yinc;
putpixel(x,y,4);
  }
getch();
}
```

3.2.1. Bresenham's ~~Ray~~ Line Algorithm

Advantages of DDA Algorithm:

i) It is the simplest algorithm and it does not require special skills for implementation.

ii) It is faster method for calculating pixel position than the direct use of line equation, $y = mx + b$.

iii) It allows us to detect the change in the value of x and y, so ~~plotting~~ plotting of same point twice is not possible.

iv) This method gives overflow indication when a point is repositioned.

v) It is easy method because it involves just two additions.

Disadvantages

i) Rounding operations and floating-point arithmetic are still time consuming.

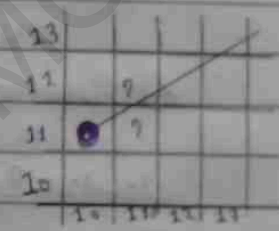
ii) The algorithm is orientation ~~depend~~ dependent. So end point accuracy is poor.

iii) The points generated by this algorithm are not accurate.

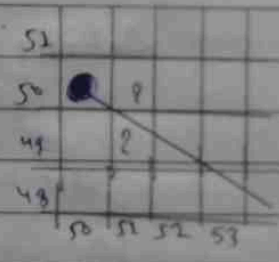
3.3.2 Bresenham's Line Algorithm

It is an accurate and efficient raster line generation algorithm, developed by J.E. Bresenham in 1962. This algorithm is used for scan converting a line. The scan converts lines only using incremental integer calculations (i.e. addition, subtraction ~~& multiplication~~)

Ideal: Move across the x-axis in unit interval and at each step choose between two y-coordinates.



Eg:
Which pixel to draw?
- (11, 11) or (11, 12) ?
- (51, 50) or (51, 49) ?



Solution: → Choice is that which is closer to the original line.

Line with positive slope less than 1 ($0 < m < 1$)

- For $|m| < 1$
  $\Delta x = 1$
- Start from left end point (x_0, y_0) step to each successive column (x samples) and plot the pixel whose scan line y value is closest to the line path.

- After (x_k, y_k) the next choice could be $(x_k + 1, y_k)$ or (x_{k+1}, y_{k+1})

y- coordinate on the mathematical line at x_{k+1} is:

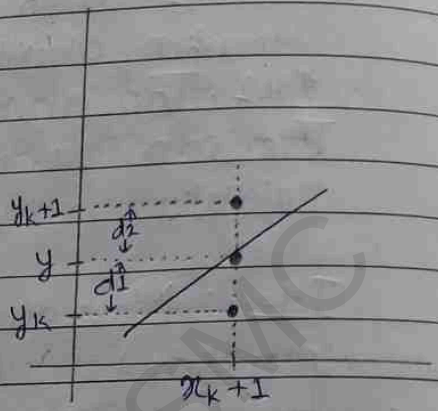
$$y = m(x_k + 1) + b$$

Then,

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

And

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



Difference between separations

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

Defining decision parameter

Let us substitute $m = \frac{dy}{dx}$ where dx and dy are the differences between the end-points.

So, a decision parameter P_k for the kth step along a line is given by,

$$P_k = dx(d_1 - d_2) \quad [\text{or } \Delta x(d_1 - d_2)]$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

[$c = 2\Delta y + \Delta x(2b - 1)$ which is independent of pixel position.]

Sign of P_k is same as that of $d_1 - d_2$ for $\Delta x > 0$ (left to right sampling)

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

$$P_{k+1} - P_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$ [c is eliminated here.]

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

So, if p_k is negative [$y_{k+1} - y_k = 0$ if $p_k < 0$; $y_{k+1} - y_k = 1$ if $p_k \geq 0$]

$y_{k+1} = y_k$ so

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

Otherwise

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

For Recursive calculation, initially

$p_0 = 2\Delta y - \Delta x$ [substitute $b = y_0 - mx_0$ and $m = \Delta y / \Delta x$]

Algorithm steps

Step 1: Input two points (x_1, y_1) and (x_2, y_2)

Step 2: Calculate $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$

Step 3: ~~If $(x_2 > x_1)$~~ Calculate decision parameter p_k.

i.e $p_k = 2\Delta y - \Delta x$

Step 4: Suppose current point is (x_k, y_k) and the next point is (x_{k+1}, y_{k+1}).

If $p_k < 0$,

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

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

~~If p_k~~

If $p_k >= 0$,

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

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

Step 5: Keep repeating step 4 until the end point is reached or number of iterations equals to $(\Delta x - 1)$ times.

Problems

1. Calculate the points between the starting coordinates (9,18) and ending coordinate (14,22).

Soln:

Given,

$(x_1, y_1) = (9, 18)$ Assume $(x_k, y_k) = (9, 18)$

$(x_2, y_2) = (14, 22)$

Calculating Δx, Δy and slope

$\Delta x = x_2 - x_1 = 5$

$\Delta y = y_2 - y_1 = 4$

$m = \Delta y / \Delta x = 0.8$

Calculate decision parameter p_k.

i.e.

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

$= 2 \times 4 - 5 = 3$

$\therefore p_k = 3$

Since, $p_k >= 0$, so we use

- $p_{k+1} = p_k + 2\Delta y - 2\Delta x = 3 + (2 \times 4) - (2 \times 5) = 1$
- $x_{k+1} = x_k + 1 = 9 + 1 = 10$
- $y_{k+1} = y_k + 1 = 18 + 1 = 19$

Similarly, the above step is repeated until the end point is reached or number of iterations equal to 4 times.

Number of iterations = $\Delta x - 1 = 5 - 1 = 4$

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

C Program for Bresenham's Line Algorithm:

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
  void main()
  {
int gd=DETECT, gm, i;
int x, y, xk, yk, x1, y1, x2, y2, p0, p, dx, dy;
initgraph(&gd, &gm, "..\\bgi");

printf("Enter co-ordinates of first point:");
scanf("%d%d", &x1, &y1);
printf("Enter co-ordinates of second point:");
scanf("%d%d", &x2, &y2);
```

```
dx = x2 - x1;
dy = y2 - y1;
x = x1;
y = y1;
Po = 2*dy - dx;
line(350, 200, 350, 200); line(350, 200, 450, 200);
while (i < dx)
{
if (Po <= 0)
{
x = x + 1;
p = Po + 2 * dy;
}
else
y = y + 1;
x = x + 1
p = Po + 2*dy - 2*dx;
putpixel (350 + x, 200 - y, 2);
Po = p;
i++;
}
getch();
}
```

Advantages of Bresenham's Line Algorithm

i) It involves only integer arithmetic, so its simple.
ii) It avoids the generation of duplicate points.
iii) It uses fixed points only.
iv) It is faster as compared to DDA.

Disadvantages of Bresenham's Line Algorithm

i) This algorithm is for basic line drawing.
ii) The result line is not smooth.

	DDA Algorithm	Bresenham's Line Algorithm
1.	DDA uses floating point i.e. Real Arithmetic.	1 Bresenham's Algorithm uses fixed point i.e. Integer Arithmetic.
2.	It uses multiplication & division as its operation.	2 It uses only addition and subtraction operations.
3.	It is slower.	3. It is faster.
4.	Not accurate and efficient.	4 More accurate and efficient.
5.		

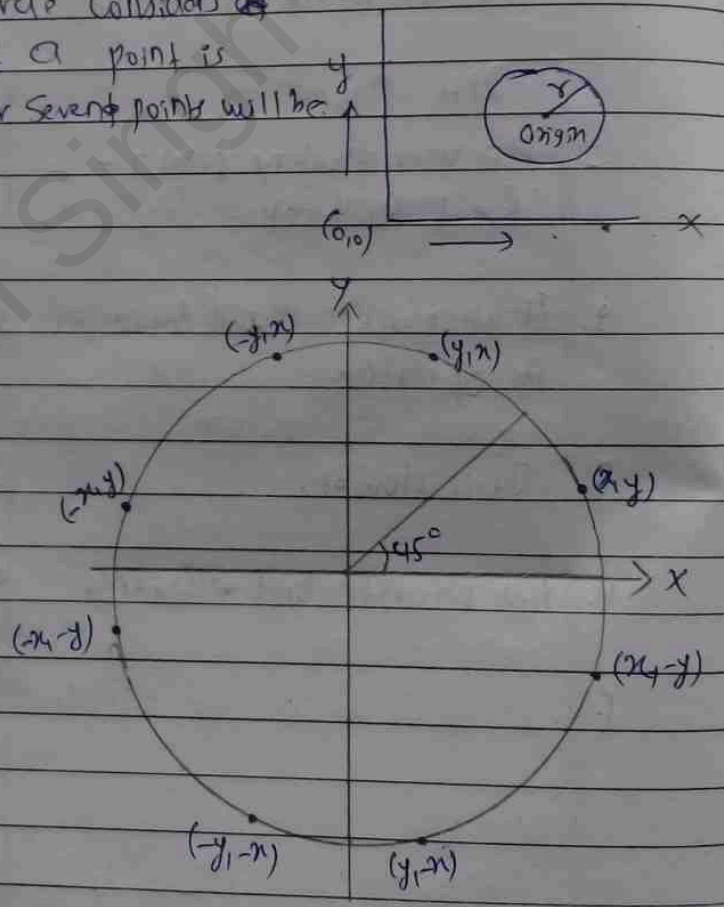
3.3. Circle Generating Algorithms: Properties of Circle, Midpoint Circle Drawing Algorithm, Bresenhams Circle drawing algorithm

Circle

Circle is an eight-way symmetric figure. The shape of circle is same in all ~~directions~~ quadrants. If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.

For drawing, circle considers ~~at~~ it at the origin. If a point is $P_1(x,y)$, then other seven points will be:

$P_2(y,x)$
$P_3(-y,x)$
$P_4(-x,y)$
$P_5(-x,-y)$
$P_6(-y,-x)$
$P_7(y,-x)$
$P_8(x,-y)$



So we will calculate only 45° arc, from which the whole circle can be determined easily.

If we want to display the circle on the screen then the putpixel function is used for eight points as shown below:

putpixel(x, y, color)
putpixel(x, -y, color)
putpixel(-x, y, color)
putpixel(-x, -y, color)
putpixel(y, x, color)
putpixel(y, -x, color)
putpixel(-y, x, color)
putpixel(-y, -x, color)

Properties of Circle

i) A circle is the set of points that are all at the same distance 'r' from a center point (x_c, y_c).

ii) The equation for a circle is expressed by Pythagorean equation in Cartesian coordinates as:

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

iii) Any given circle can be drawn on the origin (0,0) and then moved to its actual position by:
- Each calculated 'x' on the circumference moved to $x + x_c$.
- Each calculated 'y' on the circumference is moved to $y + y_c$.

Midpoint Circle Drawing Algorithm

The mid-point circle algorithm is an algorithm used to determine the points needed for rasterizing a circle. Given the centre point and radius of circle, Mid-point Circle Drawing Algorithm attempts to generate the points of one octant. The points for other octants are generated using the eight symmetry property.

In this algorithm decision parameter is based on a circle equation. As we know that the equation of circle is $x^2+y^2=r^2$ when the centre is $(0,0)$.

Now let us define the function of a circle

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

i) If $f_{circle} < 0$, then (x,y) inside the circle
ii) If $f_{circle} = 0$, then (x,y) is on the circle.
iii) If $f_{circle} > 0$, then (x,y) is outside the circle.

Algorithm

Given,

Centre point of circle $= (x_0, y_0)$

Radius $= R$

The point generation using mid point circle involves the following steps:

Step 1: Assign the starting point of the coordinates (x_0, y_0) as, $x_0 = 0$

$y_0 = R$

$\therefore (x_0, y_0) = (0, r)$

Step 2: Calculate the value of initial decision parameter P_0 as.

$P_0 = 1 - r$ (or $5/4 - r$)

Step 3: Suppose the current point is (x_k, y_k) and the next point is (x_{k+1}, y_{k+1})

Find the next point of the first octant depending on the value of decision parameter P_k.

i) if $P_k < 0$

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

$P_{k+1} = P_k + 2(x_{k+1}) + 1$

ii) if $P_k >= 0$

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

~~$P_{k+1} = P_k + 2(y_{k+1})$~~

$P_{k+1} = P_k + 2(x_{k+1}) + 1 - 2(y_{k+1})$

Step 4: Determine the symmetry points in other seven octants.

Step 5: Move at each point by the given centre (x_c, y_c) and plot the coordinate values:

$x = x + x_c$

$y = y + y_c$

Step 6: Repeat steps 3 to 5 until the condition $x >= y$.

C Program for Midpoint Circle Drawing Algorithm

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
    void pixel(int xc, int yc, int x, int y);
    int main()
    {
    int gd = DETECT, gm, xc, yc, r, x, y, p;
    detectgraph (&gd, &gm);
    initgraph (&gd, &gm, "C:\\Turbo C3\\BGI");
printf("Enter center of circle: ");
scanf("%d%d", &xc, &yc);
printf("Enter radius of circle:");
scanf("%d", &r);
x = 0;
y = r;
p = 1-r;
pixel(xc, yc, x, y);
while (x<y)
{
if(p<0)
  {
    x++;
    p = p+2*x+1;
  }
else
  {
    x++;
    y--;
    p = p+2*(x-y)+1;
```

```
    }
    pixel(xc, yc, x, y)
  }
  getch();
  closegraph();
  return 0;
  }

  void pixel(int xc, int yc, int x, int y)
  {
  putpixel(xc+x, yc+y, white);
  putpixel(xc+x, yc+y, white);
  putpixel(xc+x, yc+y, white);
  putpixel(xc+x, yc-y, white);
  putpixel(xc-x, yc+y, white);
  putpixel(xc-x, yc-y, white);
  putpixel(xc+y, yc+x, white);
  putpixel(xc-y, yc+x, white);
  putpixel(xc+y, yc-x, white);
  putpixel(xc-y, yc-x, white);
  }
```

Advantages of Midpoint Circle Drawing Algorithm

i) It is powerful and efficient algorithm.

ii) The entire algorithm is based on the simple equation of circle $x^2+y^2=R^2$.

iii) It is easy to implement from the programmer's perspective.

Disadvantages

i) Accuracy of the generating point is an issue in this algorithm.

ii) The circle generated by this algorithm is not smooth.

iii) This algorithm is time consuming.

Bresenham's Circle Drawing Algorithm

Bresenham's circle drawing algorithm is a circle drawing algorithm that selects the nearest pixel position to complete the arc. The unique part of this algorithm is that it uses only integer arithmetic which makes it significantly faster than other algorithms using floating point arithmetic in classical processors.

Given the centre point and radius of circle, Bresenham's circle Drawing Algorithm attempts to generate the points of one octant. The points for other other octants are generated using the eight symmetry property.

Algorithm

Given,

- Centre point of circle $= (x_c, y_c)$
- Radius $= r$

Step 1: Get the radius of circle r and coordinates of centre of circle (x_c, y_c).

Step 2: x and y are going to be plotted points
set $x = 0, y = r$
$\therefore (x, y) = (0, r)$

Step 3: Calculate the initial decision parameter
$d = 3 - 2r$

Step 4: Plot the circle (x_c, y_c, x, y)

Step 5: if $d < 0$ then
$d = d + 4x + 6$
$x = x + 1$
$y = y$
else
$d = d + 4(x - y) + 10$
$x = x + 1$
$y = y - 1$

Step 6: Check if $x <= y$
Go to step 7
else
Go to step 4

Step 7: Stop / End.

C - Program for Bresenham's Circle Drawing Algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#
   void pixel(int xc, int yc, int x, int y)
   int main()
   {
int gd, gm, xc, yc, r, x, y, p;
   detectgraph(&gd, &gm);
initgraph(&gd, &gm, "C://Turbo3//BGI");
printf("Enter centre of circle");
```

```
scanf("%d%d", &xc, &yc);
printf("Enter radius of circle:");
scanf("%d", &r);
x=0;
y=r;
p=3-2*r;
pixel(xc, yc, x, y);
while (x<=y)
{
if(p<0)
{
x++;
p=p+4*x+6;
x=x+1; }
else
{
x++; p=p+4*(x-y)+10;
y--; x=x+1;
y=y-1;
}
pixel (xc, yc, x, y);
}
getch();
closegraph();
return 0;
}
void pixel(int xc, int yc, int x, int y)
{
putpixel (xc+x, yc+y, 7);
putpixel(xc+x, yc-y, 7);
putpixel(xc-x, yc+y, 7);
```

putpixel(xc+x, yc+y, 7);
putpixel(xc+y, yc+x, 7);
putpixel(xc+y, yc-x, 7);
putpixel(xc-y, yc+x, 7);
putpixel(xc-y, yc-x, 7);
}

Advantages of Bresenham's Circle Drawing Algorithm:

i) The entire algorithm is based on the simple equation of circle $x^2+y^2=R^2$.

ii) It is easy to implement.

Disadvantages

i) Accuracy is an issue.
ii) Difficult to generate complex and high graphical images.

3.1 Attributes: Line Attributes, Curve attributes and Character attributes

The way a primitive is to be displayed is referred to as an Attribute Parameter.
Some attribute parameters include color, size etc.
Some types of attributes are explained below:

1. Line Attributes

Basic attributes of a straight line segment are its type, its width and its color. In some graphics packages, lines can also be displayed using selected pen or brush options.

i) Line Type

Possible selection of line type attribute includes:

- Solid lines
- Dashed lines → ~~very short dash~~
- Dotted lines
- Dash Dotted lines

To set line type attribute in PHIGS application program, a user invokes the function:

`SetLinetype (lt)`

lt can be - 1, 2, 3, 4

ii) Line Width

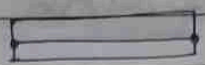
Implementation of line width option depends on the capabilities of the output device to ~~set~~ set the line width attributes.

`SetLinewidthScaleFactor (lw)`

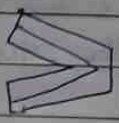
linewidth parameter lw is assigned to a positive number to indicate the relative width of line to be

displayed.
lw Value = 1, Standard line
lw value = 0.5, half of standard line
lw value > 1, thicker than standard line

Line cap Endcaps

- Butt Cap 
- Round Cap 
- Projecting Square Cap 

Line Joins

- Mitter join 
- Roundjoin 
- Bevel join 

iii) Line Color

A polyline routine displays a line in the current color by setting the color value in the framebuffer at pixel locations along the line path using the set pixel procedure.

`SetPolyLineColourIndex(lc)`

A line drawn with background colour is invisible.

iv) Pen and Brush Options

With some packages, lines can be displayed with pen or brush selections. Options in this category includes:

: shape

size

pattern

2. Curve Attributes

Parameters for curve attributes are same as those for line segments. Curves displayed with varying colors, widths, dot-dash patterns and available pen or brush options.

3. Character Attributes

The appearance of displayed character is controlled by attributes such as font, size, color and orientation.

Attributes can be set for both entire character strings (text) and for individual characters defined as marker symbols.

i) Text Attributes

SetTextFont (tf)

SetTextColourIndex (tc)

SetCharacterHeight (ch)

SetCharacterExpansionFactor (cw)

SetCharacterSpacing (cs)

SetCharacterUpVector (upvect)

SetTextPath (tp)

SetTextAlignment (h,v)

SetTextPrecision (tpr)

Unit- 4

2-D Geometric Transformations

Transformation is the process of modifying and re-positioning the existing graphics.

Some graphics are changed into something else by applying some of the rules, known as Transformation. Various types of transformations are there such as translation, rotation, scaling, reflection, shearing etc. The transformation when takes place in 2D plane, is known as 2D Transformation.

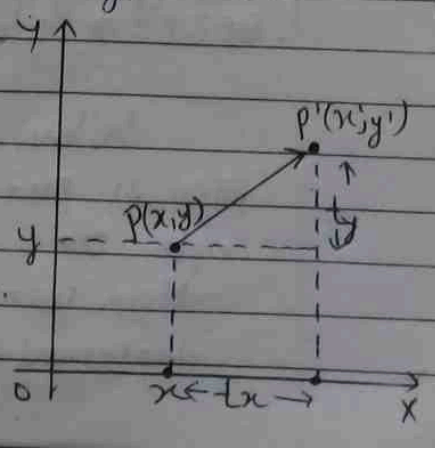
In order to reposition the graphics on the screen and change the size of orientation, transformations play a crucial role in computer graphics.

4.1. Basic Transformations : Translation, Rotation, Scaling

1. Translation

Translation is the straight line movement of an object from one position to another. It is called Translation. Here the object is positioned from one coordinate location to another.

To translate a point in 2D from coordinate position (x, y) to new coordinate (x', y'), we add translation distances or coordinates (t_x, t_y) to the original coordinate.



From the figure,

$x' = x + t_x$

$y' = y + t_y$

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

The above equations can also be represented using column vectors as:

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

So, we can write.

$$P' = P + T$$

2. Rotation

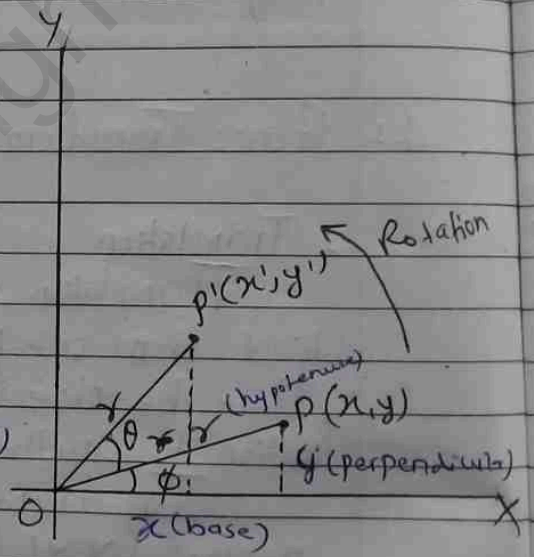
It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise.

Consider a point object $P(x,y)$ in XY-plane which has to be rotated from one angle to another in a 2D plane.

From the figure.

Let,

- Initial Coordinates of $P(x,y)$
- Coordinates after rotation $= P'(x',y')$
- Initial angle of object P with respect to origin $= \phi$
- Rotation angle $= \theta$
- $OP = r$ (Constant distance from origin)



When OP is rotated through an angle 'θ' taking origin as pivot point for rotation, then OP' makes an angle $\phi + \theta$ with X-axis.

Now, using trigonometric rules, $P(x,y)$ can be represented as.

$\cos\phi = \frac{b}{h} = \frac{x}{r}$ | $\sin\phi = \frac{p}{h} = \frac{y}{r}$

or, $x = r\cos\phi$ — ① | or, $y = r\sin\phi$ — ②

Same way $P(x', y')$ is represented as:

$\cos(\phi+\theta) = \frac{x'}{r}$ | $\sin(\phi+\theta) = \frac{y'}{r}$

or, $x' = r\cos(\phi+\theta)$ — ③ | $\therefore y' = r\sin(\phi+\theta)$ — ④

$= r\cos\phi\cos\theta - r\sin\phi\sin\theta$ — ③ | $= r\cos\phi\sin\theta + r\sin\phi\cdot\cos\theta$ — ④

Substituting ① & ② in ③ & ④ respectively, we get

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

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

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

$\therefore P' = RP$

or, $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ — Matrix form

This is clockwise notation (negative notation)

If anticlockwise notation, then (positive notation) then,

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

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

or, $P' = P \times R$

Q. Rotate point P(4,3) in anticlockwise by 45°

Soln:

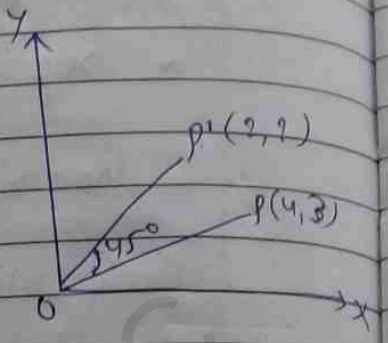
We have

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

$$R = \begin{bmatrix} \cos 45° & \sin 45° \\ -\sin 45° & \cos 45° \end{bmatrix}$$

or,

$$R = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$



$$\therefore P' = P \cdot R$$

$$= \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$= [4, 3] \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} 4/\sqrt{2} - 3/\sqrt{2} & 4/\sqrt{2} + 3/\sqrt{2} \end{bmatrix}$$

$$P' = \begin{bmatrix} 1/\sqrt{2} & 7/\sqrt{2} \end{bmatrix}$$

$$\therefore P(x', y') = (1/\sqrt{2}, 7/\sqrt{2})$$

3. Scaling

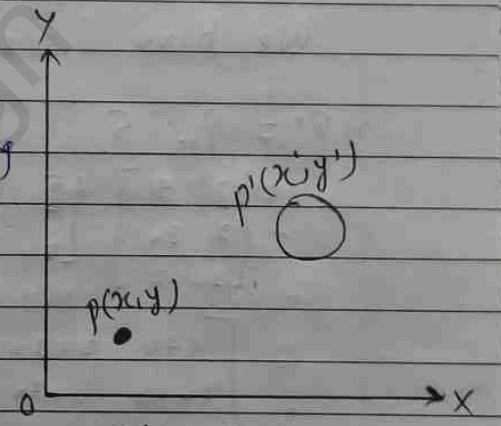
Scaling is a process of modifying or altering the size of objects. The change is done using the scaling factors. There are two scaling factors i.e. S_x in x-direction and S_y in Y-direction.

- If Scaling factor > 1, then the object size is increased.
- If Scaling factor < 1, then the object size is reduced.

- If S_x and S_y are equal, it is also called Uniform scaling.
- If S_x and S_y are not equal it is called differential scaling.

Consider a point object P(x, y) has to be scaled using scaling factors (S_x, S_y) to produce P'(x', y').

This can be mathematically represented as:



$$x' = x \cdot S_x \text{ and } y' = y \cdot S_y.$$

It can also be represented in matrix form as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$ where S is Sca

or, $P' = P \times S$ where, S = scaling matrix.

Q. Scale the polygon A(2,5), B(7,10) and C(10,2) by 2 units in x-direction and 2 units in y direction.

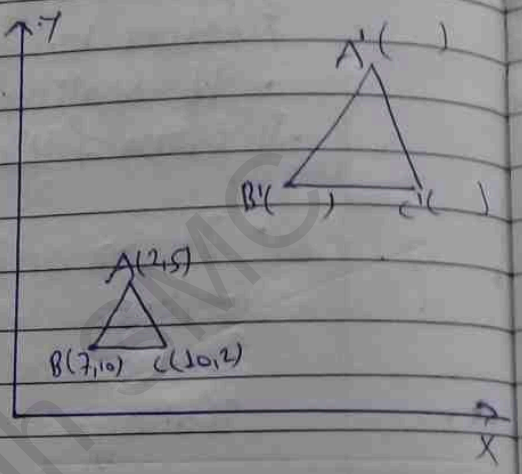
Soln:

Given,

$S_x = 2, S_y = 2$

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$P = \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix}$$



We have,

$$P' = P \cdot S$$

$$= \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 4+0 & 0+10 \\ 14+0 & 0+20 \\ 20+0 & 0+4 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 10 \\ 14 & 20 \\ 20 & 4 \end{bmatrix}$$

$\therefore A' = (4, 10)$

$B' = (14, 20)$

$C' = (20, 4)$

4.2 Composite Transformations : Translation, Scaling, Rotation

A number of transformations can be combined into single one called Composite transformation. The resulting matrix is called composite matrix. The process of combining is called Concatenation.

Mathematically,

When the transformation of plane T_1 is followed by a second plane transformation T_2, then the resultant single transformation T which is the composition of T_1 and T_2 taken in that order. This is written as, $\boxed{T = T_1 \cdot T_2}$

Composite transformation can be achieved by Concatenation of transformation matrices to obtain a combined transformation matrix.

A combined matrix,

$$[T][X] = [X][T_1][T_2][T_3] \ldots [T_n]$$

Where, $[T_i]$ is any combination of:

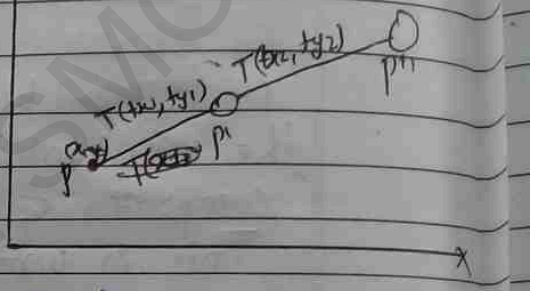
- Translation
- Rotation
- Scaling
- Reflection
- Shearing

1. Translation

Two successive translations are Additive.

If two successive translation vectors (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}) are applied to a coordinate position P, the final translation location P' is calculated as

$$P' = T(t_{x2}, t_{y2}) \cdot \{T(t_{x1}, t_{y1}) \cdot P\}$$
$$= \{T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1})\} \cdot P$$



Where P and P' are represented as homogeneous-coordinate column vectors. We can verify this by calculating the matrix product for the two associative groupings.

Also, the composite transformation matrix for this sequence of translation is:

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

or, $T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$

OR

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

2. Rotation

Two successive rotation operations are additive.

Let $R(\theta_1)$ and $R(\theta_2)$ be two successive rotations applied to a coordinate position P produce the transformed position.

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

Here the composite transformation matrix for this sequence of rotation is

$$R(\theta_2) \cdot R(\theta_1) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \times \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

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

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

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

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

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

OR Rotation,

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

3. Scaling

Two successive scaling operations are multiplicative.

Let $S(s_{x1}, s_{y1})$ and (s_{x2}, s_{y2}) be two successive vector applied to a coordinate position P then the Composite scaling thus produced is:

$$P' = S(s_{x2}, s_{y2}) \cdot \{S(s_{x1}, s_{y1}) \cdot P\}$$
$$\therefore P' = \{S(s_{x2}, s_{y2}) \cdot S(s_{x1}, s_{y1})\} P.$$

The Composite transformation matrix for this sequence of scaling is:

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x2} \cdot s_{x1} & 0 & 0 \\ 0 & s_{y2} \cdot s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore S(s_{x2}, s_{y2}) \times S(s_{x1}, s_{y1}) = S(s_{x1} \cdot s_{x2}, s_{y1} \cdot s_{y2})$$

OR

$$\text{Scaling, } \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

4.3 Recent Transformation Concepts and Advantages

Significance of 2D transformation:

- Transformations are the building blocks of computer graphics.
- Positioning, shaping, viewing positions are done by transformations.
- Transformation is also used for determining views.
- The transformations that one can perform in 2 dimensions:

i) Translation → Movement of all points of an object within a fixed direction for an object is known as translation.

ii) Rotation → Changing the angle of the object either clockwise or anticlockwise.

iii) Scaling → Increasing or decreasing the coordinates of an object is scaling.

4.4. Two dimensional object to screen viewing transforms

Window → A world coordinate area selected for display is called a window. The window defines what is to be viewed.

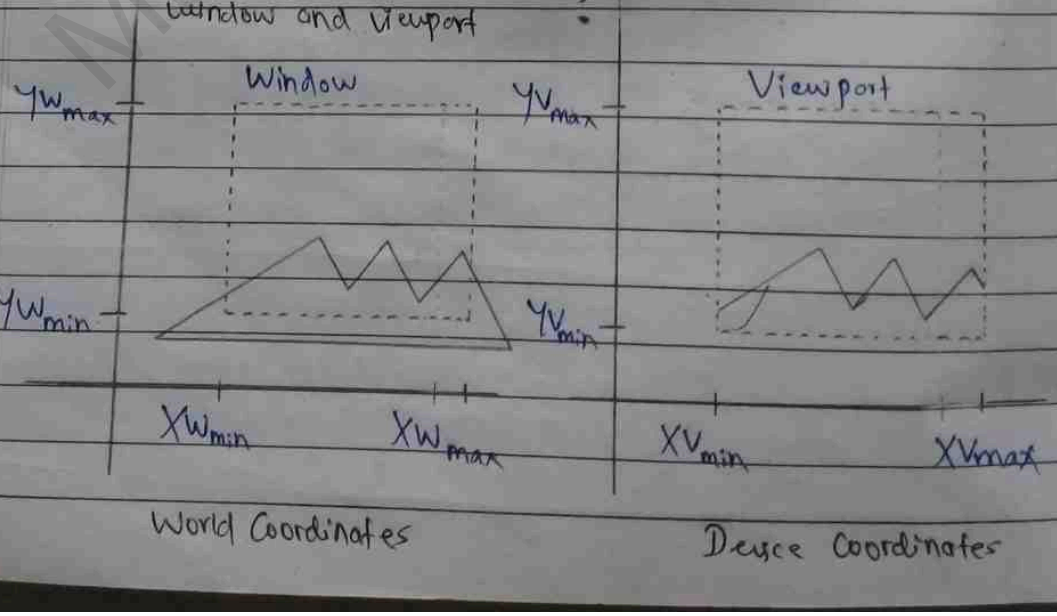
Viewport → An area on a display device to which a window is mapped is called a view port. The viewport defines where it is to be displayed.

Viewing Transformation / window-to-viewport transformation

The mapping of a part of an object from world-coordinate scene to device coordinates or screen coordinates is known as viewing transformation.

The two dimensional viewing transformation is referred to as window to viewport transformation or windowing transformation.

Fig: A viewing transformation using standard rectangles for the window and viewport



2D- Viewing Transformation Pipeline (or steps):

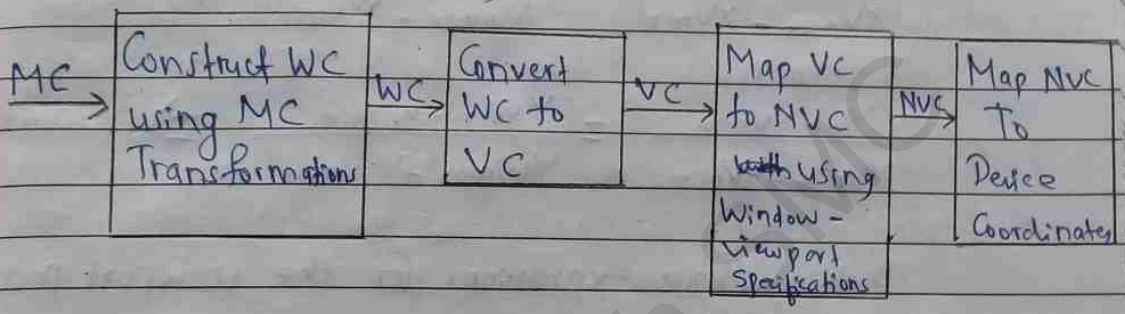
We should know,

MC = Modelling Coordinates

WC = World Coordinates

VC = Viewing Coordinates

NVC = Normalized Viewing Coordinates



Window to Viewport Coordinate Transformation:

Window-to-viewport Coordinate transformation is done by maintaining the same relative placement of objects in device coordinates as they had been in world coordinates. For example, if a coordinate position is at the centre of the viewing window, then it will be displayed at the centre of the viewport.

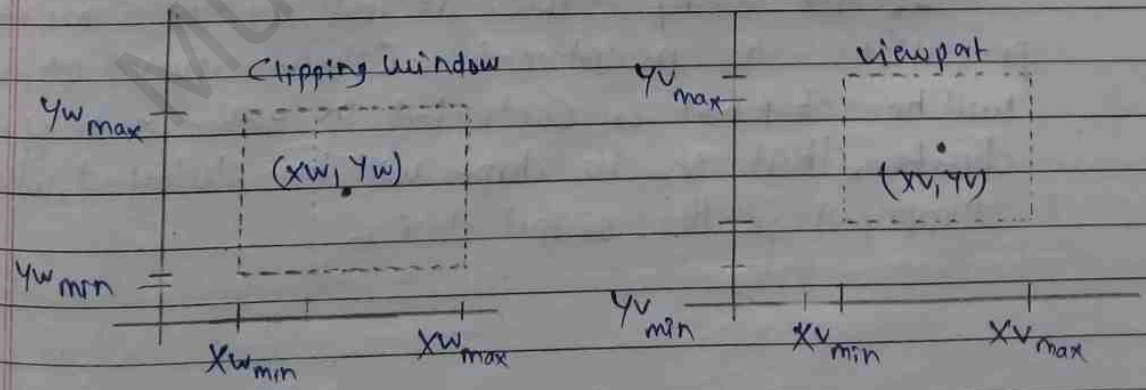


Fig: A point at position (xw, yw) in a window is mapped to viewport coordinates (xv, yv) so that the relative positions in the two areas are the same.

A point at position (x_w, y_w) in window is mapped to position (x_v, y_v) in the viewport. To maintain the same relative placement in viewport, we require:

$$\frac{x_v - x_{v_{min}}}{x_{v_{max}} - x_{v_{min}}} = \frac{x_w - x_{w_{min}}}{x_{w_{max}} - x_{w_{min}}} \Rightarrow \text{x-coordinate}$$

and

$$\frac{y_v - y_{v_{min}}}{y_{v_{max}} - y_{v_{min}}} = \frac{y_w - y_{w_{min}}}{y_{w_{max}} - y_{w_{min}}} \Rightarrow \text{y-coordinate}$$

Solving these expressions for the viewport position (x_v, y_v), we have

$$x_v = x_{v_{min}} + (x_w - x_{w_{min}})\, s_x$$

and $$y_v = y_{v_{min}} + (y_w - y_{w_{min}}) \cdot s_y$$

Where the scaling factors (s_x, s_y) are:

$$s_x = \frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}, \quad s_y = \frac{y_{v_{max}} - y_{v_{min}}}{y_{w_{max}} - y_{w_{min}}}$$

If the scaling factors, s_x and s_y are same, relative proportions are maintained. Otherwise, world object will be stretched or contracted in either x or y direction, that is, its shape will be distorted, when displayed on the output device.

4.5 Workstation Transformation

The workstation transformation defines that the part of NDC space which will be visible, and where it will appear on the display surface.

Any number of output device can be open in particular application and another window viewport transformation can be performed for each open output device. This mapping is called the workstation transformation, is accomplished by selecting a window area in normalized space and a viewport area in coordinates of display device.

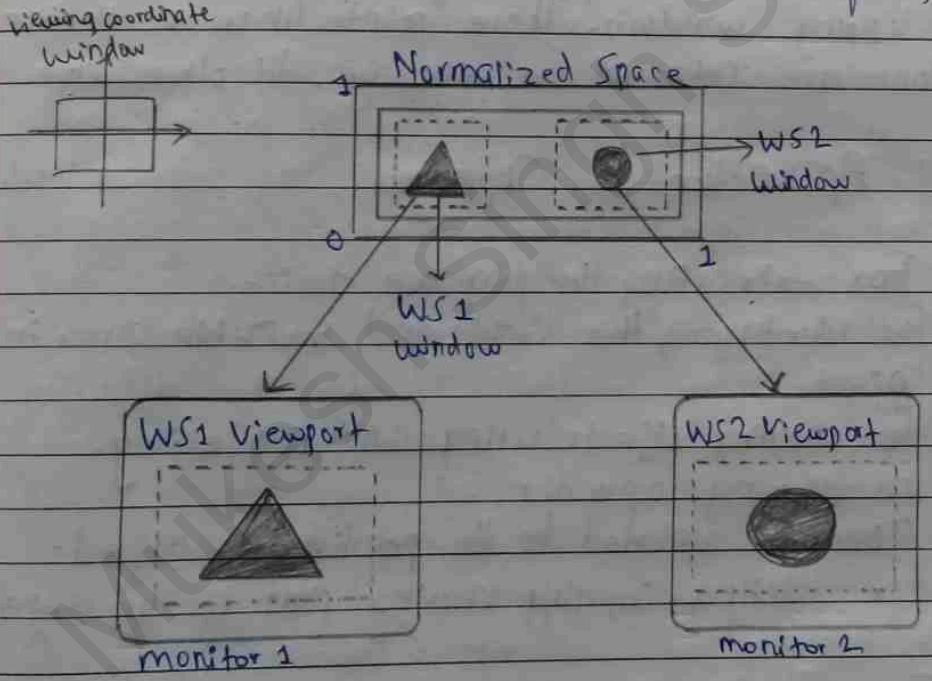


Fig: Mapping selected parts of a scene in normalized coordinate to different video monitors with workstation transformation.

4.6 Clipping Operations: Point Clipping, Line Clipping, Cohen-Sutherland Line Clipping, Liang-Barsky Line Clipping

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

. The clipping algorithm determines which points, lines or portion of lines lie within the clipping window. These points, lines or portions of lines are retained for display. All other are discarded.

Applications of Clipping:

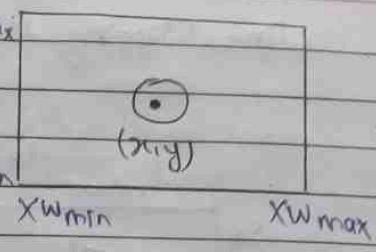
- For extracting the part we desire.
- For identifying the visible and invisible area in 3D object.
- For creating objects using solid modelling.
- For drawing operations.
- Operations related to the pointing of object.
- For deleting, copying, moving part of an object.

Types of Clipping:

1. Point Clipping

Clip window is a rectangle in standard form. Point Clipping is used to determine, whether the point is inside the window or not. For this following conditions are checked:

1. $x \leq xw_{max}$
2. $x \geq xw_{min}$
3. $y \leq yw_{max}$
4. $y \geq yw_{min}$



The point (x,y) is for display. If anyone of from the above inequalities is false, then the point will fall outside that window and will not be considered to be visible.

2. Line Clipping:

A line clipping procedure involves several parts:

i) First we test the given line segment whether it lies completely inside the clipping window.

ii) If it does not, we try to determine whether it lies completely outside the window.

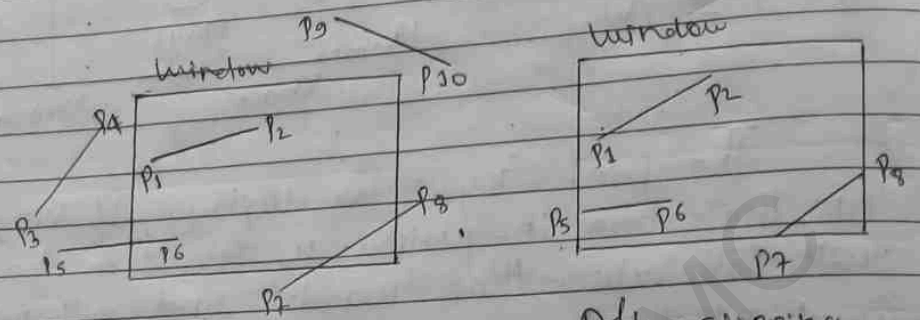
iii) Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries.

Process the lines through "inside-outside" tests by checking the line endpoints.

Inside-Outside Test

- Check both endpoints, -> If P_1 & P_2 are within boundaries - line is saved.
- If both endpoints outside, drop line segment.
- Remaining may require calculation of multiple intersection -s.

Line-Clipping against a rectangular clip window:



Befor Clipping After clipping

All other lines cross one or more clipping boundaries. For a line segment with endpoints (x_1, y_1) and (x_2, y_2) and one or both endpoints outside the clipping ~~window~~ rectangle, the parametric representation.

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

could be used to determine the values of parameter u for intersection with the clipping boundary coordinates.

3. Cohen-Sutherland line Clipping

It is developed by Dan Cohen and Ivan Sutherland. To speed up the processing this algorithm performs initial tests that reduces the number of intersections that must be calculated.

In this algorithm, it is detected whether line lies inside the screen or it is outside the screen. All lines comes under any one of the

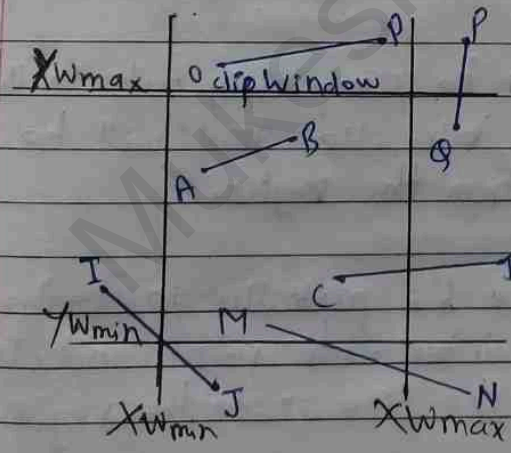
following categories:

1. Visible – If a line lies within the window, a line is visible and will be displayed as it is.

2. Not visible – If a line lies outside the window, it will be invisible and rejected. Such lines will not display.

3. Clipping case – lines can be partial, inside window then we should find intersection point and draw only that inside region.

This algorithm uses the clipping window as shown in the figure below. The minimum coordinate for the clipping region is (XWmin, YWmin) and the maximum coordinate for the clipping region is (XWmax, YWmax).



- Line AB is visible case.
- Line OP is invisible case.
- Line PQ is invisible case.
- Line IJ are clipping candidates.
- Line MN are clipping candidates.
- Line CD are clipping candidates.

The entire region is divided into by using 4-bits which represent Top, Bottom, Right and Left of region. There are 9 regions and all are assigned

codes of 4-bits. If both endpoints of the line have end bits zero '0', then the line is considered to be visible.

region 1	region 2	region 3
region 4	region 5	region 6
region 7	region 8	region 9

fig:- 9 regions

Top (TBRL)

1001	1000	1010
0001	0000	0010
0101	0100	0110

fig - Bits assigned to 9 regions

There are 3 different possibilities for the line:

i) Visible Case - line can be completely inside the window (This line is visible and should be accepted).

ii) Invisible Case - line can be completely outside the window. (This line is invisible and will be completely removed from the region.

iii) Clipping Case - line can be partially inside the window. (We will find intersection point and draw only that portion of the line inside the region.

Algorithm for Cohen-Sutherland line Clipping

Step 1: Assign a region code for each endpoints.
Step 2: Calculate the positions of endpoints of the line

Step 3: Perform OR operation on both of these end-points.

Step 4: If OR operation gives 0000
then, line is visible and accepted.
else,
Perform AND operation on both end-points.
If AND ≠ 0000
Then, the line is invisible and rejected.
else if AND = 0000
line is considered the clipped case.

Step 5: If a line is clipped case, find an intersection with boundaries of the window.
$m = (y_2 - y_1)/(x_2 - x_1)$

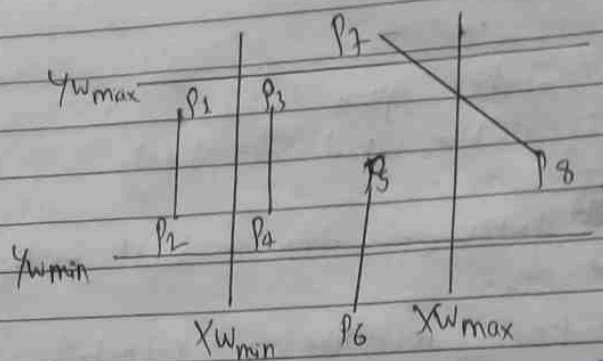
(a) If bit 1 is "1", line intersects with left boundary of rectangle window.
$y_3 = y_1 + m(x - x_1)$, where $x = x_{wmin}$

(b) If bit 2 is "1", line intersects with right boundary
$y_3 = y_1 + m(x - x_1)$, where $x = x_{wmax}$

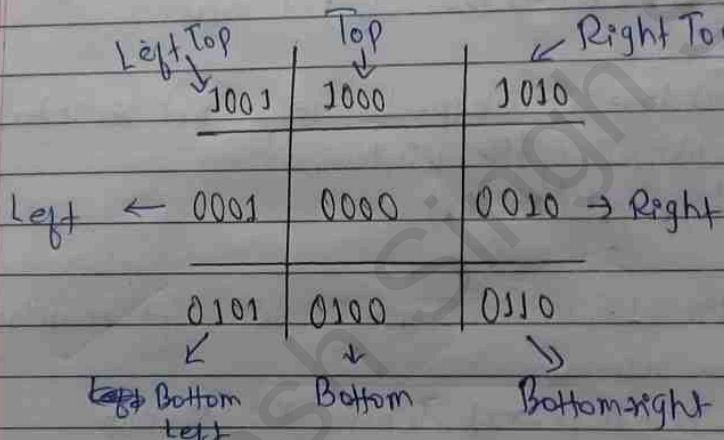
(c) If bit 3 is "1", line intersects with bottom boundary.
$x_3 = x_1 + (y - y_1)/m$, where $y = y_{wmin}$

(d) If bit 4 is "1", line intersects with top boundary
$x_3 = x_1 + (y - y_1)/m$, where $y = y_{wmax}$

Cohen-Sutherland Example:



We know that in Cohen-Sutherland,



Step 1: P_1 & P_2

$P_1 = 0001$ → Non-zero value

$P_2 = 0001$ → Non-zero value

Using AND operation.

P_1 AND $P_2 = 0001$ → Non zero

The line is completely outside the window.

Step 2: P_3 & P_4

$P_3 = 0000$ → Zero

$P_4 = 0000$ → Zero

Line is accepted because it is completely inside the window.

Step 3: P5 & P6

P5 = 0000 → zero value

P6 = 0100 → non-zero value

P5 AND P6 = 0000 → zero value

The line is partially inside the window and partially outside the window, so clipping is required.

P5 = 0000
P6' = 0000
P5 AND P6' = 0000

Step 4: P7 & P8

P7 = 1000 → non-zero

P8 = 0010 → non-zero

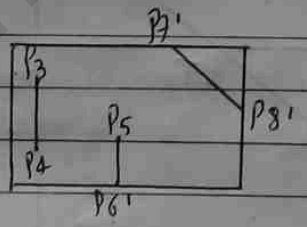
P7 AND P8 = 0000 → zero

The line is partially inside the window, so it needs clipping.

P7' = 0000

P8' = 0000

P7' AND P8' = 0000



After clipping

Advantages of Cohen-Sutherland Line Clipping:

i) Calculates end-points very quickly.

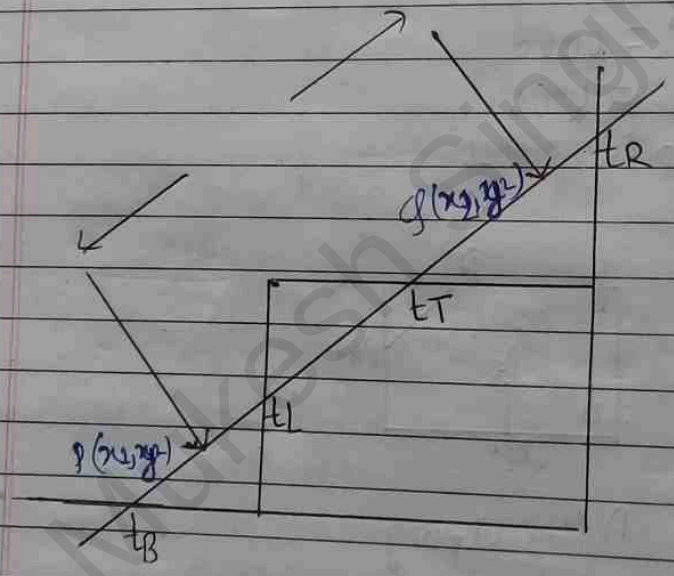
ii) Rejects and accepts lines quickly.

iii) Can clip pictures much larger than screen size.

4. Liang-Barsky Line Clipping

Liang and Barsky have established an algorithm that uses floating point arithmetic but finds the appropriate endpoints with at most four computations. This algorithm uses the parametric equation for a line and solves four inequalities to find the range of the parameter for which the line is in the viewport.

It is more efficient than Cohen-Sutherland line clipping and can be extended to 3d clipping.



Let $P(x_1, y_1)$ and $Q(x_2, y_2)$ be the line which we want to study. The parametric equation of the line is given by:

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

$$y = y_1 + t(y_2 - y_1) = y_1 + dy * t$$

(Where, t is between 0 and 1)

We can see that when $t = 0$, the point computed

If $P(x_1, y_1)$ and when $t=1$, the point computed is $Q(x_2, y_2)$.

Writing the point clipping conditions in parametric form:

$$xw_{min} \le x_1 + t(x_2 - x_1) \le xw_{max}$$
$$yw_{min} \le y_1 + t(y_2 - y_1) \le yw_{max}$$

Algorithm for Liang-Barsky line Clipping Algorithm:

Step 1: Accept the line endpoints $A(x_1, y_1)$ and $B(x_2, y_2)$ and window coordinates $Xw_{min}, Yw_{min}, Xw_{max}, Yw_{max}$ as input.

Step 2: Calculate p_k and q_k for $k = 1, 2, 3, 4$ as

$p_1 = -\Delta x$	$q_1 = x_1 - xw_{min}$	($p_1, q_1 \rightarrow$ left)
$p_2 = \Delta x$	$q_2 = xw_{max} - 1$	($p_2, q_2 \rightarrow$ right)
$p_3 = -\Delta y$	$q_3 = y_1 - yw_{min}$	($p_3, q_3 \rightarrow$ bottom)
$p_4 = \Delta y$	$q_4 = yw_{max} - y_1$	($p_4, q_4 \rightarrow$ top)

Step 3: If $p_k = 0$ then line is parallel to kth boundary
if $q_k < 0$ then line is outside the boundary, discard the line and stop.
If $q_k \ge 0$ then line is inside the boundary, calculate intersection points.
if $p_k \ne 0$ then go to step 4.

~~Step 4: Calculate $t_k = \frac{q_k}{p_k}$ for $k = 1, 2, 3, 4$ (q_1, q_2, q_3, q_4)~~

Step 4: If $p_k < 0$, find t_1 initial time

$$t_1 = \max\left(0, \frac{q_k}{p_k}\right) \quad [x = x_1 + t_1 \Delta x, y = y_1 + t_1 \Delta y]$$

If $p_k > 0$, find t_2 final time

$t_2 = \min(1, \frac{q_k}{p_k})$ $\left[\begin{matrix} x = x_1 + t_2 \Delta x \\ y = y_1 + t_2 \Delta y \end{matrix}\right]$

Step 5: If $t_1 > t_2$ then line is totally outside, discard it and stop.

If $t_1 < t_2$ then calculate endpoints of clipped line.

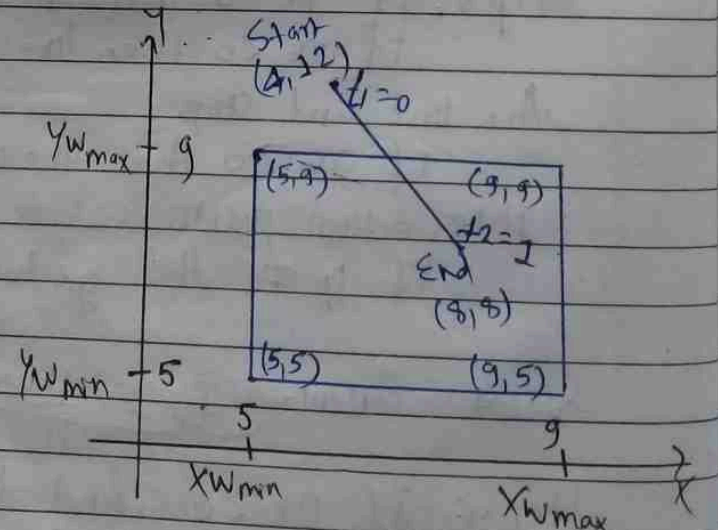
$x' = x_1 + t_1 \Delta x \quad y' = y_1 + t_1 \Delta y \quad I'(x', y')$

$x'' = x_1 + t_2 \Delta x \quad y'' = y_1 + t_2 \Delta y \quad I''(x'', y'')$

Step 6: Display line segment from I' to I''.

Example

Consider the windowsize from 5 to 9. Clip the following line A(4,12) and B(8,8)



Soln:

1. Given end points are: $A(x_1, y_1) = (4, 12)$

$B(x_2, y_2) = (8, 8)$

Window Coordinates are:

$X_{Wmin} = 5$

$X_{Wmax} = 9$

$Y_{Wmin} = 5$

$Y_{Wmax} = 9$

$\Delta x = x_2 - x_1 = 4$

$\Delta y = y_2 - y_1 = -4$

2. Calculate P_k and q_k for $k = 1, 2, 3, 4$.

$P_1 = -\Delta x = -4$, $q_1 = x_1 - x_{Wmin} = 4 - 5 = -1$

$P_2 = \Delta x = 4$, $q_2 = x_{Wmax} - x_1 = 9 - 4 = 5$

$P_3 = -\Delta y = 4$, $q_3 = y_1 - y_{Wmin} = 12 - 5 = 7$

$P_4 = \Delta y = -4$, $q_4 = y_{Wmax} - y_1 = 9 - 12 = -3$

3. Calculate t_1 and t_2

$P_1, P_4 < 0$

$t_1 = \max\left(0, \frac{q_k}{p_k}\right)$

$= \max\left(0, \frac{q_1}{p_1}, \frac{q_4}{p_4}\right)$

$= \max\left(0, \frac{1}{4}, \frac{3}{4}\right)$

$t_1 = \frac{3}{4}$

$P_2, P_3 > 0$

$t_2 = \min\left(1, \frac{q_k}{p_k}\right)$

$= \min\left(1, \frac{q_2}{p_2}, \frac{q_3}{p_3}\right)$

$= \min\left(1, \frac{5}{4}, \frac{7}{4}\right)$

$t_2 = 1$

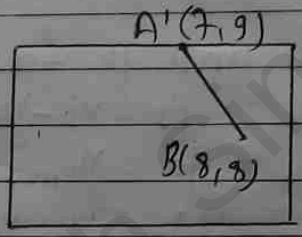
4. Compare t_1 and t_2 and find end points of clipped line.

Since $t_1 < t_2$

$x = x_1 + t_1 \Delta x = 4 + \frac{3}{4} \times 4 = 7$

$y = y_1 + t_1 \Delta y = 12 + \frac{3}{4} \times (-4) = 12 - 3 = 9$

$\therefore (x, y) = (7, 9)$



Clipped line

Unit-5

Three-Dimensional Graphics

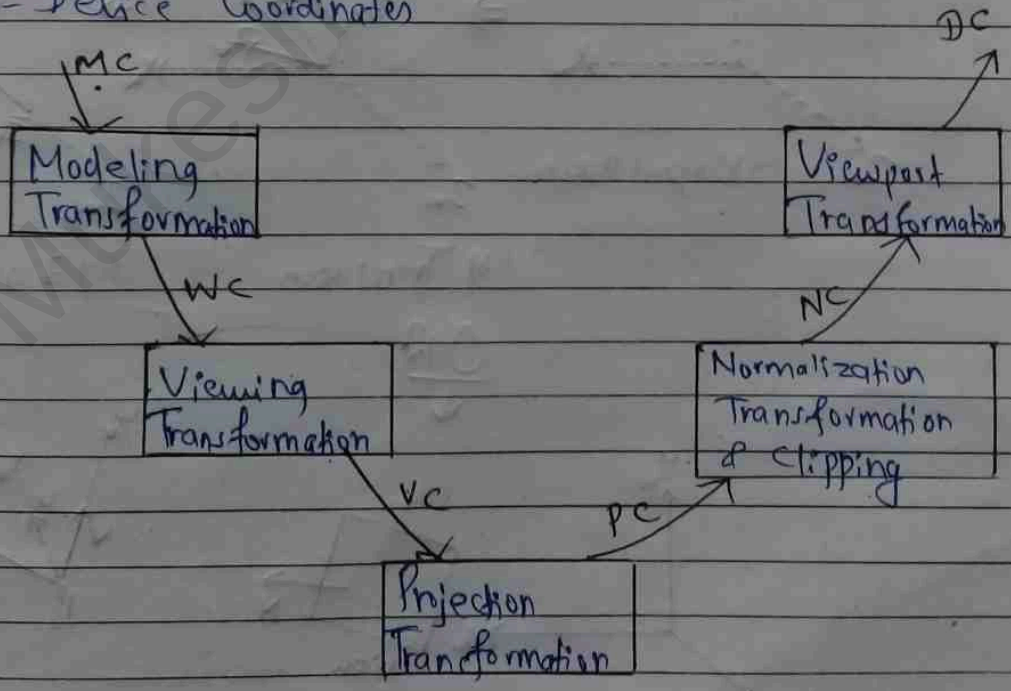
Three-dimensional (3D) computer graphics (CG), are graphics that use a three dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. The resulting images may be ~~later~~ stored for viewing later or displayed in real time.

5.1. Three-dimensional viewing pipeline

We know,

- MC = Modeling Coordinates
- WC = World Coordinates
- VC = Viewing Coordinates
- PC = Projection Coordinates
- NC = Normalized Coordinates
- DC = Device Coordinates

3D → Process → 2D Devices

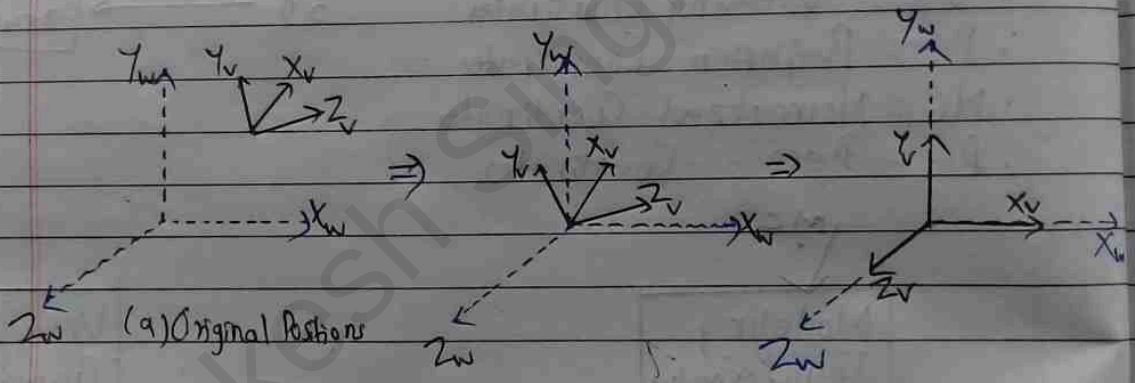


Transformation from world Coordinates to Viewing Coordinates:

The conversion of object description from world coordinates to viewing coordinates is achieved by following transformation sequence:

1) Translate the viewing-coordinate origin to the world coordinate origin.

2) Apply rotations to align the u, v, n axes with the world ~~X~~ Coordinates X_w, Y_w, Z_w axes respectively.

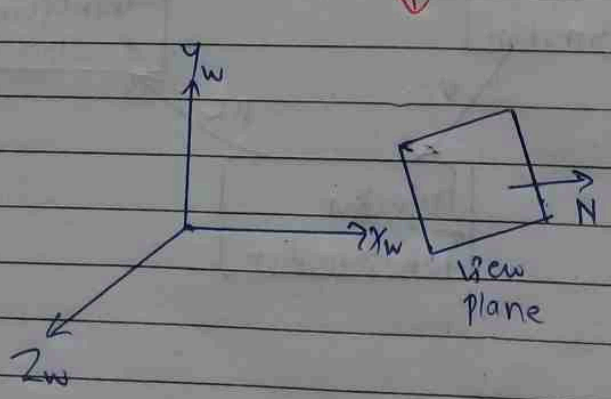


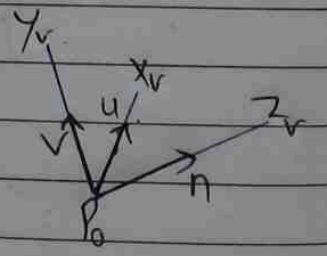
(a) Original Positions

(b) Translation

(c) 3 axes rotation

OR





$$R = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{WC,VC} = R \cdot T = \begin{bmatrix} u_x & u_y & u_z & -u \cdot P_0 \\ v_x & v_y & v_z & -v \cdot P_0 \\ n_x & n_y & n_z & -n \cdot P_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

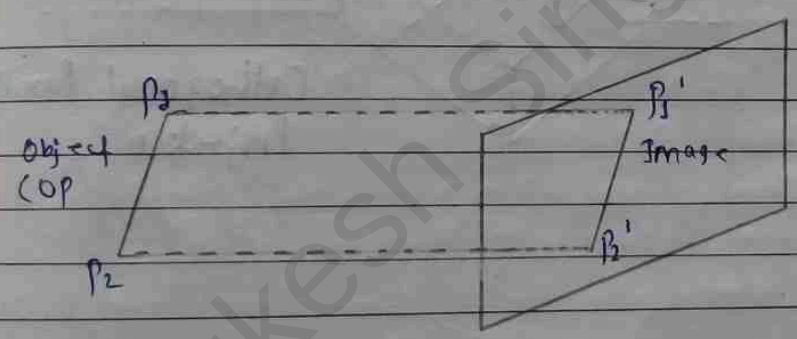
5.2 Projection: Parallel Projections, Perspective Projections

The technique used for 2-D displays of 3-D objects is called projection.
In geometry, several types of projections are known. However, in Computer graphics, mainly two i) the parallel and ii) the perspective projections are of interest.

~~1. Parallel Projection~~:

1. Parallel Projection

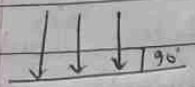
- Mostly used by drafters and engineers to create working drawings of an object which preserves its scale and shape.
- In parallel projection, Z coordinate is discarded and parallel line from each vertex on the object are ~~intersected to each other~~ extended until they intersect the view plane.
- The point of intersection is the projection of the vertex.
- The distance between the COP (Centre of Projection) and the projection plane (PP) is infinite i.e. the projectors are parallel to each other and have a fixed direction.
- Coordinate positions are transformed to the view plane along parallel lines.

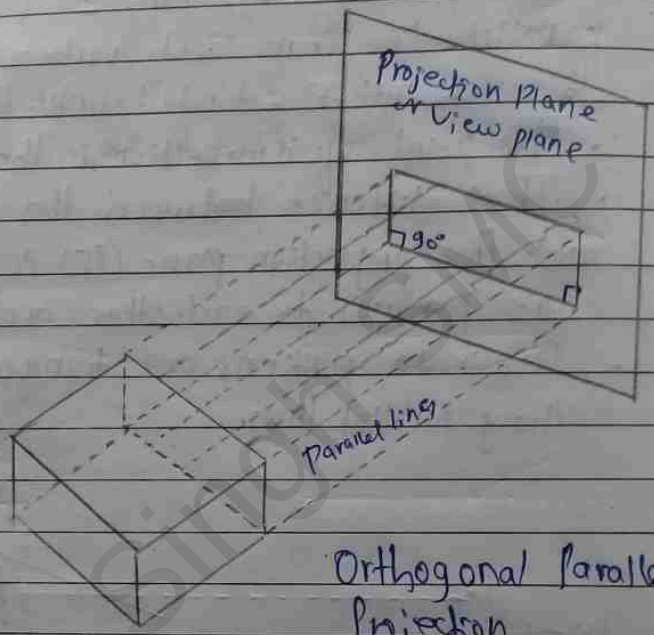


i) Orthographic Parallel Projection

- The ~~projection~~ direction is perpendicular to the view plane.
- The centre of projection is infinity.
- Parallelism of lines is preserved but angles are not.
- The orthographic projection can display more than one face of an object called axonometric orthographic projection.
- Most commonly used axonometric orthographic projection

is the Isometric projection.
. Lines are parallel to each other. making an angle of 90° with the view plane.

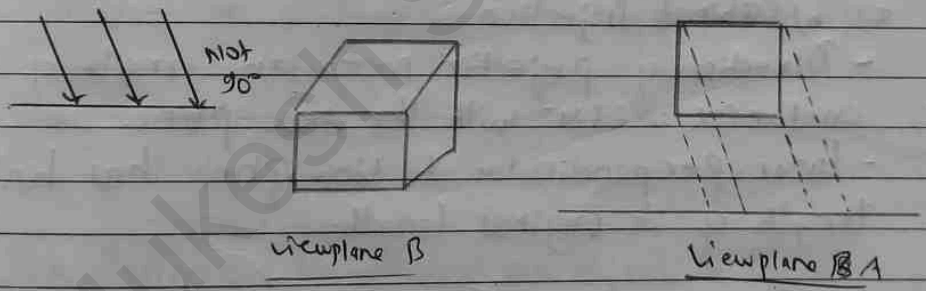




Orthogonal Parallel Projection

ii) Oblique Parallel Projection

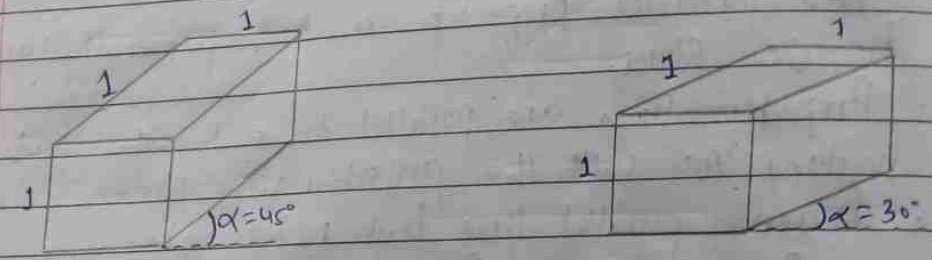
- The parallel projection is not perpendicular to the view plane.
- Projection lines are parallel to each other but not making 90° with the projection/view plane.
- Preserve parallel lines but not angles.
- - Isometric view: Projection plane is placed symmetrically with respect to the three principal faces that meet at a corner of object.
  - Dimetric View: Symmetric with two faces.
  - Trimetric View: General Case
- Direction of Projection (DOP) is perpendicular to the projection/view plane.
- Only faces of the object parallel to the projection plane are shown true size and shape.



Two Common oblique parallel projections:

a) Cavalier Projection

- Direction of projection makes an angle of 45° with the view plane.
- Projection of line perpendicular to the view plane has the same length as the original length.



(a)

Fig: Cavalier Projection of the Unit Cube

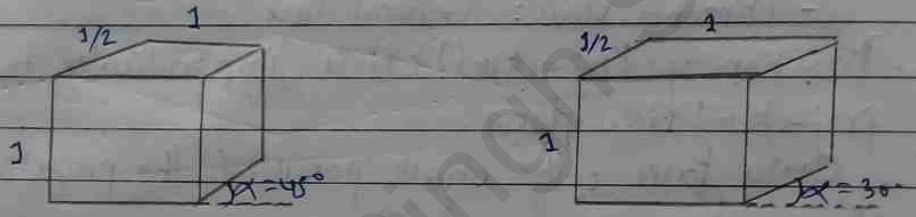


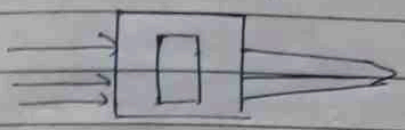
Fig: Cabinet Projection of the Unit Cube

b) Cabinet Projection

- Direction of projection makes an angle of $\arctan(2) = 63.4°$ with the view plane.
- Lines perpendicular to view plane has half length of its original length.

2. Perspective Projections

- First discovered by Donatello, Brunelleschi and Da Vinci during Renaissance.
- Objects closer to viewer look larger.
- Parallel lines appear to converge to single point.
- Lines of projection are not parallel. Instead they all converge at a single point called the Centre of projection or projection reference point.



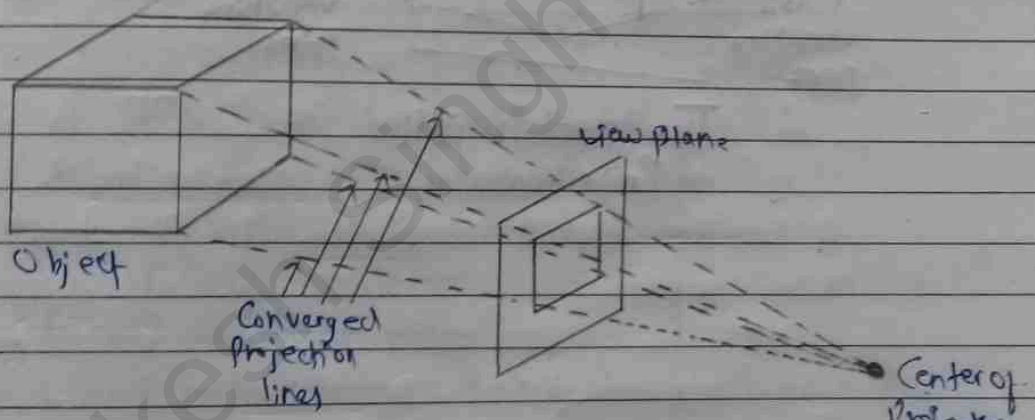
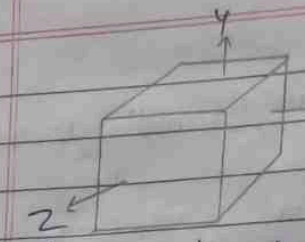


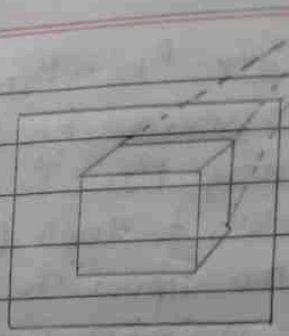
Fig: Perspective projection of an object to the view plane.

There are 3 types of perspective projections:

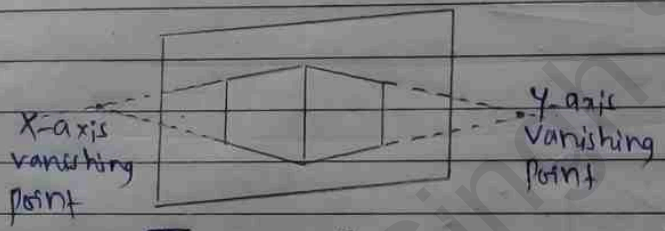
i) One-Point Perspective Projection
ii) Two-Point Perspective Projection
iii) Three-point perspective projection



Coordinate description



One-point Perspective Projection



Two-point Perspective Projection

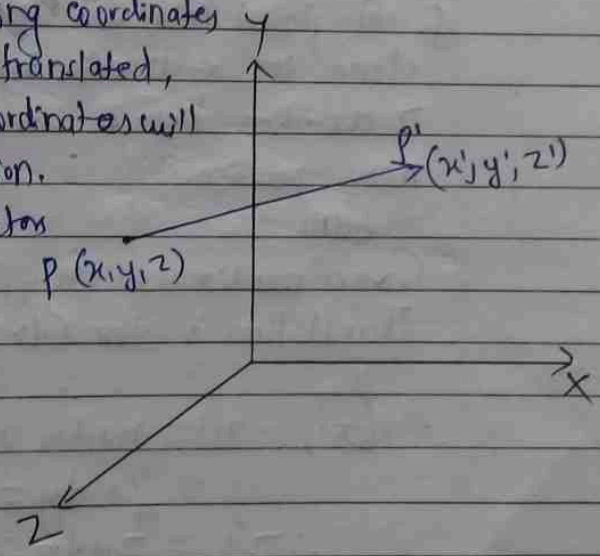
5.3. Extension of two-dimensional transforms to three-dimensions: Translation, Rotation, Scaling

The three-dimensional transformations are extensions of two-dimensional transformation. In 2D two coordinates x and y are used whereas in 3D coordinates x, y and z are used.

For three-dimensional images and objects, three dimensional transformations are needed. These are translation, rotation and scaling. These are also called basic transformations are represented using matrix. More complex transformations are handled using matrix in 3D.

1. Translation – Movement of obj. from one position to another.

If P is a point having coordinates in three directions (x, y, z) is translated, then after translation, its coordinates will be (x', y', z') after translation.
t_x, t_y, t_z are translation vectors in x, y, and z directions respectively.



Then,

$x' = x + t_x$

$y' = y + t_y$

$z' = z + t_z$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$ in column vector form.

$$\therefore P' = P + T$$

Representing translation using matrix:

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

OR

$$[x', y', z', 1] = [x \; y \; z \; 1] \times \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore [x' \; y' \; z' \; 1] = [x + t_x, \; y + t_y \; \; z + t_z \; \; 1]$$

Q. A point has coordinates (5, 6, 7). The translation is done in x-direction by 3, y-direction by 3 and z-direction by 2 coordinates. Shift the object find the new position.

Soln:

Given coordinates $(x, y, z) = (5, 6, 7)$

Translation vector $(t_x, t_y, t_z) = (3, 3, 2)$

$\therefore x' = x + t_x = 5 + 3 = 8$

$\therefore y' = y + t_y = 6 + 3 = 9$

$\therefore z' = z + t_z = 7 + 2 = 9$

~~$\therefore (x, y, z) = (5$~~

So, after translation,

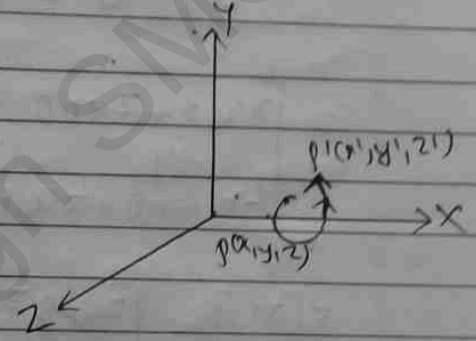
$(x', y', z') = (8, 9, 9)$.

2. Rotation

It is moving an object about an angle. Movement can be anticlockwise or clockwise. For 2D we describe the angle of rotation, but for a 3D we angle of rotation and axes of rotation are required. The axes can be either x, y or z.

i) Rotation about x-axis anticlockwise

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



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

$$= \begin{bmatrix} 1\times x + 0\times y + 0\times z \\ 0\times x + y\times\cos\theta - z\times\sin\theta \\ 0\times x + y\times\sin\theta + z\times\cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y\cos\theta - z\sin\theta \\ y\sin\theta + z\cos\theta \end{bmatrix}$$

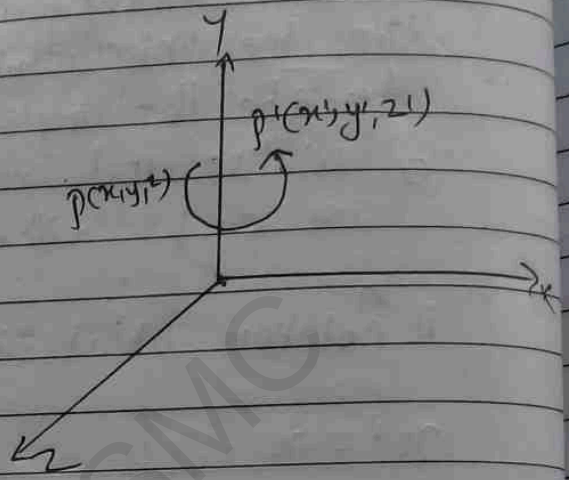
OR

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

OR

$$R = \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

ii) Rotation about y-axis anticlockwise



$y' = y$

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

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

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

$$= \begin{bmatrix} x\cos\theta + z\sin\theta \\ y \\ -x\sin\theta + z\cos\theta \end{bmatrix}$$

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

OR

$$R_y(\theta) = \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

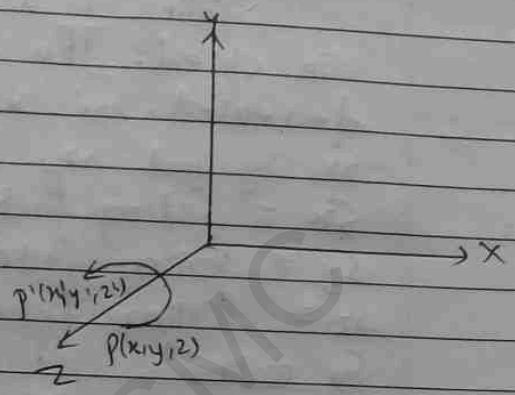
(iii) Rotation about z-axis anticlockwise

$z' = z$

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

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

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



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

OR

$$R_z(\theta) = \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

3 Scaling

Scaling is used to change the size of an object. The size can be increased or decreased. For scaling 3 factors are required S_x, S_y and S_z.

S_x = Scaling factor in x-direction

S_y = Scaling factor in y-direction

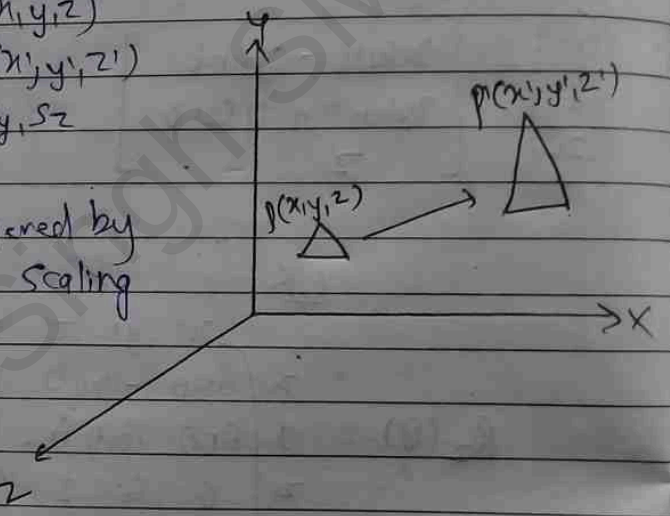
S_z = Scaling factor in z-direction

For scaling

Initial Coordinates = $P(x, y, z)$

Final Coordinates = $P'(x', y', z')$

Scaling factors = S_x, S_y, S_z



The scaling is achieved by using the following scaling equations:

$x' = x \cdot S_x$

$y' = y \cdot S_y$

$z' = z \cdot S_z$

In matrix form

$P' = P \cdot S$

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

$$= \begin{bmatrix} S_x x + 0 \times y + 0 \times z \\ 0 \times x + S_y \times y + 0 \times z \\ 0 \times x + 0 \times y + S_z \times z \end{bmatrix}$$

$$\therefore \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \cdot S_x \\ y \cdot S_y \\ z \cdot S_z \end{bmatrix}$$

OR

$$P' = P \cdot S$$

$$\therefore [x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \cdot S$$

$$= [x \; y \; z \; 1] \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$\therefore [x' \; y' \; z' \; 1] = [x \cdot S_x \; y \cdot S_y \; z \cdot S_z \; 1]$$

5.5 Hidden line and hidden surface removal techniques

When a picture that contains the non transparent objects and surfaces are viewed, the objects that are behind the objects that are closer cannot be viewed. To obtain a realistic screen image, these hidden surfaces need to be removed. This process of identification and removal of these surfaces is known as Hidden surface problem.

The hidden surface problems can be solved by two methods - Object space method and Image-space method.

In physical coordinate system, object-space method is implemented and in case of screen coordinate system, image-space method is implemented.

These methods are also called as Visible Surface Determination.

1.) Object-space method

In this method, various parts of objects are compared. After comparison, visible, invisible or hardly visible surface is determined. These method generally decide visible surface. In the wireframe model, these are used to determine a visible line. So these algorithms are line based instead of surface based.

2) Image Space Methods

Here, positions of various pixels are determined. It is used to locate the visible surface instead of a visible line. Each point is detected for its visibility. If a point is visible then the pixel is on, otherwise off.

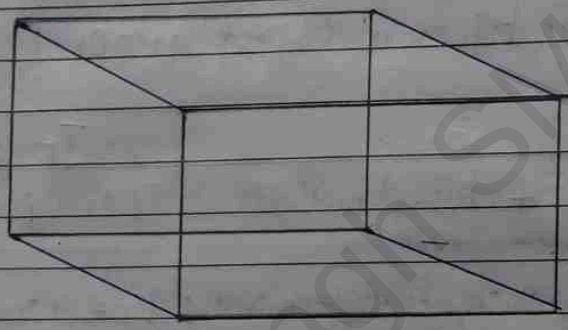
Differences between Object Space and Image Space methods:

Object Space	Image Space
1. It is object based method.	1. It is a pixel-based method.
2. Surface visibility is determined.	2. Line a point visibility is determined.
3. Calculation are not based on the resolution of the display.	4. Calculations are resolution based.
4. These are developed for vector graphics system.	4. These are developed for raster devices.
5. It requires a lot of calculations if the image is to enlarge.	5. Image can be enlarged without losing accuracy.
6. If the number of objects in the scene increases, computation time also increases.	6. In this method, complexity increases with the complexity of visible parts.

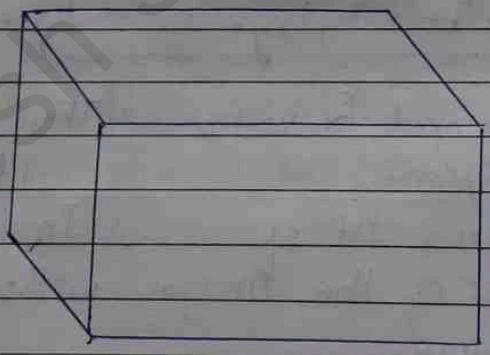
Similarity of object and Image Space method:

→ or

→ In both methods, sorting is used a depth comparison of individual lines, surfaces are objected to their distances from the view plane.



Object with hidden line



Object when hidden lines removed

Algorithm

Algorithms wed for hidden line surface detection

1. Back face removal Algorithm
2. Z-Buffer Algorithm
3. Painter Algorithm
4. Scaline line Algorithm
5. Sub division Algorithm
6. Floating horizon Algorithm

5.6. Need for shading in data visualization

Shading

Shading is a way to paint the object with light.

Shading is referred to as the implementation of the illumination model at the pixel points or polygon surfaces of the graphics objects.

Shading model is used to compute the intensities and colors to display the surface. The shading model has two primary ingredients: properties of the surface and properties of the illumination falling on it. The principle surface property is its reflection, which determines how much of the incident light is reflected. If a surface has different rectangle for the light of different wavelengths, it will appear to be coloured.

Data Visualization

Data visualization is the representation of data or information in a graph, chart or other visual format.

It is important because it allows trends and patterns to be more easily seen.

Why do we need data visualization?

We need data visualization because a visual summary of information makes it

easier to identify patterns and trends than looking through thousands of rows on a spreadsheet. It's the way the human brain works.

5.7 Basic illumination models: ambient, diffuse and specular reflections

Illumination models

An illumination model, also called a lighting model and sometimes referred to as a shading model, is used to calculate the intensity of light that we should see at a given point on the surface of an object.

- Simplified and fast methods for calculating surfaces intensities, mostly empirical.
- Calculations are based on optical properties of surfaces and lighting conditions (no reflected sources nor shadows).
- Light sources are considered to be point sources.
- Reasonably good approximation for most scenes.

There are

1) Ambient Illumination model

- In this model, we can ~~simply~~ set a general level of brightness for a scene. This is a simple way to model the combination of light reflections from various surfaces to produce a uniform illumination called the ambient light or background light.

- The amount of ambient light incident on each object is constant for all surfaces and over all directions.

- Very simple model, not very realistic.

- The reflected intensity I_{amb} of any point on the surface is:

$$I_{amb} = K_a I_a$$

I_a → ambient light intensity
$K_a \in [0, 1]$ → surface ambient reflectivity

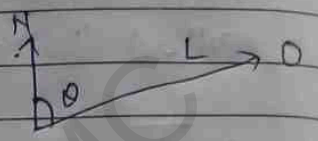
2) Diffuse Reflection Model

- Diffuse reflections are constant over each surface in a scene.

- Surfaces that are rough or grainy tend to reflect light in all directions. This scattered light is called diffuse reflection.

- The surface appears equally bright from all viewing directions.

- The brightness of each point is proportional to $\cos\theta$



- The reflected intensity I_{diff} of a point on the surface is:

$$I_{diff} = K_d I_p \cos(\theta) = K_d I_p (N.L)$$

I_p → the point light intensity.
$K_d \in [0,1]$ → the surface diffuse reflectivity

N → the surface normal
L → the light direction

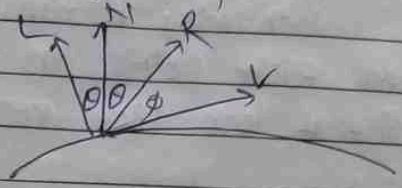
Note → If N and L have unitary length:
$\cos\theta = N.L$

3) Specular Reflection model

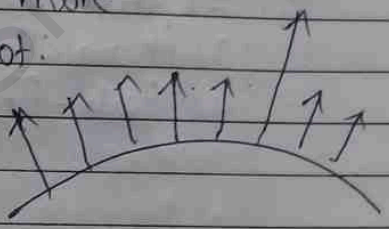
: We see ~~a bright~~ a highlight or bright spot, at certain viewing directions. This phenomenon, called Specular reflection, is the result of total or near total reflection of the incident light in a concentrated region around the

Specular reflection angle.

- The specular-reflection angle equals the angle of the incident light.



- In specular reflection it may possible that some portion of the surface generated more light or may produce bright spot.



- The Phong model: reflected specular intensity falls of as some power of Cos (ϕ):

$$I_{spec} = K_s I_p \cos^n(\phi) = K_s \cdot I_p (R.V)^n$$

K_s → the surface specular reflectivity

n → specular ~~parameter~~ reflection parameter

The illumination equation combined with all three reflections is:

$$I = I_{amb} + I_{diff} + I_{spec}$$

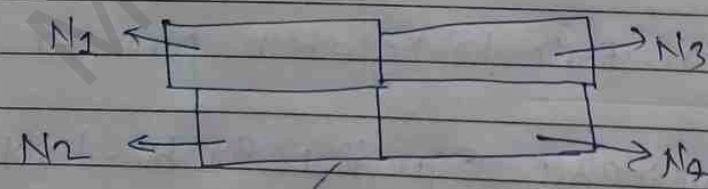
$$I = K_a I_a + I_p (K_d N.L + K_s (R.V)^n)$$

5.8 Polygon-Rendering Methods: Constant, Gouraud, Phong and Fast-Phong

Rendering means giving proper intensity at each point in a graphical object to make it look like real world object.

Different rendering methods are:
1) Constant Intensity Shading
2) Gouraud Shading
3) Phong Shading
4) Fast-Phong Shading

1) Constant Intensity Shading
- Also called flat shading.
- Fast and simple method.
- A single intensity is calculated for each polygon (in the polygon-mesh).
- All points in the surface of polygon are displayed with the same intensity value.



For getting Good results through Constant Intensity Shading:
- Object is polyhedron.
- Light sources illuminating the object are sufficiently far from the surface.

. The viewing position is sufficiently far from the surface.

2) Gouraud Shading

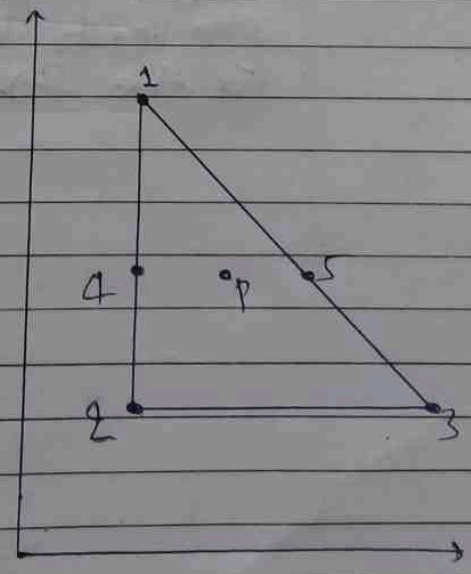
. Intensity interpolation scheme developed by Gouraud.
. Represents a polygon surface by linearly interpolating intensity across the polygon surface.
. Intensity values for each polygon are matched with adjacent polygon along common edges.
. Eliminates the intensity discontinuity that can occur in flat shading.

Polygon surface is calculated by following calculations:

i) Determine the average unit vector at each polygon vertex.

ii) Apply an illumination model to each vertex to calculate the vertex intensity.

iii) Linearly interpolate the vertex intensities over the polygon surface.

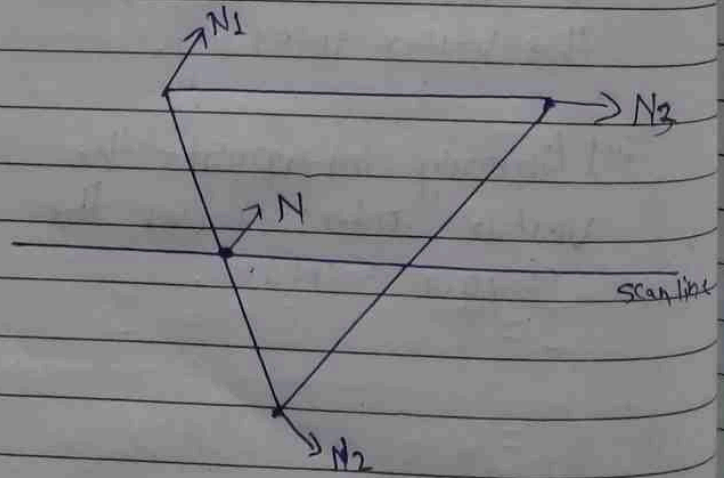


3) Phong Shading

- Normal vector interpolation shading.
- Developed by Phong Bui Tuong.
- Accurate method.
- Displays more realistic highlights.
- Reduces ~~match~~ Mach band effect.
- Interpolate normal vector and then apply illumination.

Polygon surface is rendered using the following steps:

- Determine the average vector at each polygon vertex.
- Linearly interpolate the vertex normals over the surface of the polygon.
- Apply illumination model along each ~~scanline~~ scan line to calculate projected pixel intensities for the surface points.



4) Fast-Phong Shading

- It is a goal of any one writing a 3D engine.
- Phong shading is still common in 3D system.
- It is faster and easier way to approximate the intensity calculation.
- Rendering is faster than the normal Phong shading.
- Surface ~~rendering~~ rendering can be speed up using approximations in the illumination model calculations of normal vectors.